Triage issues

The main goal of issue triage is to categorize all incoming Grafana issues and make sure each issue has all basic information needed for anyone else to understand and be able to start working on it.

> **Note:** This information is for Grafana project Maintainers, Owners, and Admins. If you are a Contributor, then you will not be able to perform most of the tasks in this topic.

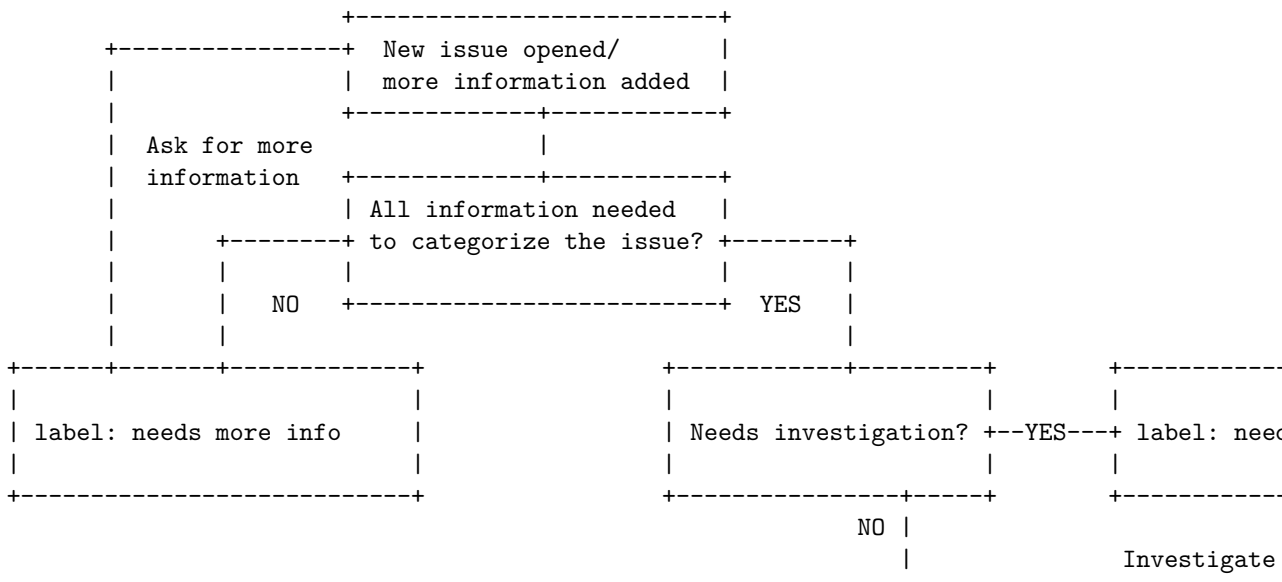
The core maintainers of the Grafana project are responsible for categorizing all incoming issues and delegating any critical or important issue to other maintainers. Currently one maintainer each week is responsible. Besides that part, triage provides an important way to contribute to an open source project.

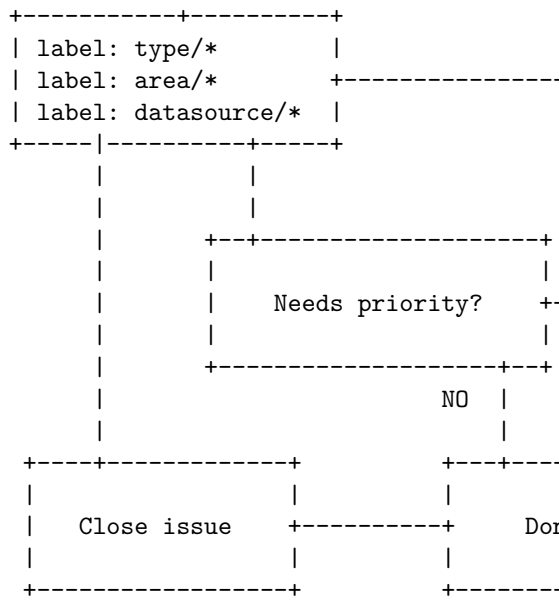
Triage helps ensure issues resolve quickly by:

- Ensuring the issue's intent and purpose is conveyed precisely. This is necessary because it can be difficult for an issue to explain how an end user experiences a problem and what actions they took.
- Giving a contributor the information they need before they commit to resolving an issue.
- Lowering the issue count by preventing duplicate issues.
- Streamlining the development process by preventing duplicate discussions.

If you don't have the knowledge or time to code, consider helping with triage. The community will thank you for saving them time by spending some of yours.

Simplified flowchart diagram of the issue triage process





1. Find uncategorized issues

To get started with issue triage and finding issues that haven't been triaged you have two alternatives.

Browse unlabeled issues

The easiest and straight forward way of getting started and finding issues that haven't been triaged is to browse unlabeled issues and starting from the bottom and working yourself to the top.

Subscribe to all notifications

The more advanced, but recommended way is to subscribe to all notifications from this repository which means that all new issues, pull requests, comments and important status changes are sent to your configured email address. Read this guide for help with setting this up.

It's highly recommended that you setup filters to automatically remove emails from the inbox and label/categorize them accordingly to make it easy for you to understand when you need to act upon a notification or where to look for finding issues that haven't been triaged etc.

Instructions for setting up filters in Gmail can be found here. Another alternative is to use Trailer or similar software.

2. Ensure the issue contains basic information

Before triaging an issue very far, make sure that the issue's author provided the standard issue information. This will help you make an educated recommendation on how to categorize the issue. The Grafana project utilizes GitHub issue templates to guide contributors to provide standard information that must be included for each type of template or type of issue.

Standard issue information that must be included

Given a certain issue template have been used by the issue author or depending how the issue is perceived by the issue triage responsible, the following should help you understand what standard issue information that must be included.

Bug reports Should explain what happened, what was expected and how to reproduce it together with any additional information that may help giving a complete picture of what happened such as screenshots, query inspector output and any environment related information that's applicable and/or maybe related to the reported problem:

- Grafana version
- Data source type & version
- Platform & OS Grafana is installed on
- User OS & Browser + versions
- Using docker + what environment
- Which plugins
- Configuration database in use (sqlite, mysql, postgres)
- Reverse proxy in front of Grafana, what version and configuration
- Non-default configuration settings
- Development environment like Go and Node versions, if applicable

Enhancement requests Should explain what enhancement or feature that the author wants to be added and why that is needed.

Accessibility issues This is a mix between a bug report and enhancement request but focused on accessibility issues to help make Grafana improve keyboard navigation, screen-reader support and being accessible to everyone. The report should include relevant WCAG criteria, if applicable.

Support requests In general, if the issue description and title is perceived as a question no more information is needed.

Good practices

To make it easier for everyone to understand and find issues they're searching for it's suggested as a general rule of thumbs to:

- Make sure that issue titles are named to explain the subject of the issue, has a correct spelling and doesn't include irrelevant information and/or sensitive information.
- Make sure that issue descriptions doesn't include irrelevant information, information from template that haven't been filled out and/or sensitive information.
- Do your best effort to change title and description or request suggested changes by adding a comment.

  Note: Above rules is applicable to both new and existing issues of the Grafana project.

Do you have all the information needed to categorize an issue?

Depending on the issue, you might not feel all this information is needed. Use your best judgement. If you cannot triage an issue using what its author provided, explain kindly to the author that they must provide the above information to clarify the problem. Label issue with `needs more detail` and add any related `area/*` or `datasource/*` labels.

If the author provides the standard information but you are still unable to triage the issue, request additional information. Do this kindly and politely because you are asking for more of the author's time.

If the author does not respond to the requested information within the timespan of a week, close the issue with a kind note stating that the author can request for the issue to be reopened when the necessary information is provided.

When you feel you have all the information needed you're ready to categorizing the issue.

If you receive a notification with additional information provided but you are not anymore on issue triage and you feel you do not have time to handle it, you should delegate it to the current person on issue triage.

3. Categorizing an issue

An issue can have multiple of the following labels. Typically, a properly categorized issue should at least have:

- One label identifying its type (`type/*`).
- One or multiple labels identifying the functional areas of interest or component (`area/*`) and/or data source (`datasource/*`), if applicable.

Label	Description
`type/bug`	A feature isn't working as expected given design or documentation.
`type/feature-request`	Request for a new feature or enhancement.

Label	Description
`type/docs`	Documentation problem or enhancement.
`type/accessibility`	Accessibility problem or enhancement.
`type/question`	Issue is a question or is perceived as such.
`type/duplicate`	An existing issue of the same subject/request have already been reported.
`type/works-as-intended`	A reported bug works as intended/by design.
`type/build-packaging`	Build or packaging problem or enhancement.
`area/*`	Subject is related to a functional area of interest or component.
`datasource/*`	Subject is related to a core data source plugin.

Duplicate issues

Make sure it's not a duplicate by searching existing issues using related terms from the issue title and description. If you think you know there is an existing issue, but can't find it, please reach out to one of the maintainers and ask for help. If you identify that the issue is a duplicate of an existing issue:

1. Add a comment `/duplicate of #<issue number>`. GitHub will recognize this and add some additional context to the issue activity.
2. The Grafana bot will do the rest, adding the correct label and closing comment
3. Optionally add any related `area/*` or `datasource/*` labels.

Bug reports

If it's not perfectly clear that it's an actual bug, quickly try to reproduce it.

It's a bug/it can be reproduced:

1. Add a comment describing detailed steps for how to reproduce it, if applicable.
2. Label the issue `type/bug` and at least one `area/*` or `datasource/*` label.
3. If you know that maintainers wont be able to put any resources into it for some time then label the issue with `help wanted` and optionally `beginner friendly` together with pointers on which code to update to fix the bug. This should signal to the community that we would appreciate any help we can get to resolve this.
4. Move on to prioritizing the issue.

It can't be reproduced:

1. Either ask for more information needed to investigate it more thoroughly.
2. Either delegate further investigations to someone else.

It works as intended/by design:

1. Kindly and politely add a comment explaining briefly why we think it works as intended and close the issue.
2. Label the issue `type/works-as-intended`.

Enhancement/feature?

1. Label the issue `type/feature-request` and at least one `area/*` or `datasource/*` label.
2. Move on to prioritizing the issue.

Documentation issue?

First, evaluate if the documentation makes sense to be included in the Grafana project:

- Is this something we want/can maintain as a project?
- Is this referring to usage of some specific integration/tool and in that case is that a popular use case in combination with Grafana?
- If unsure, kindly and politely add a comment explaining that we would need upvotes to identify that lots of other users want/need this.

Second, label the issue `type/docs` and at least one `area/*` or `datasource/*` label.

Minor typo/error/lack of information:

There's a minor typo/error/lack of information that adds a lot of confusion for users and given the amount of work is a big win to make sure fixing it:

1. Either update the documentation yourself and open a pull request.
2. Either delegate the work to someone else by assigning that person to the issue and add the issue to next major/minor milestone.

Major error/lack of information:

1. Label the issue with `help wanted` and `beginner friendly`, if applicable, to signal that we find this important to fix and we would appreciate any help we can get from the community.
2. Move on to prioritizing the issue.

Accessibility issues

1. Label the issue `type/accessibility` and at least one `area/*` or `datasource/*` label.

Support requests

1. Kindly and politely direct the issue author to the community site and explain that GitHub is mainly used for tracking bugs and feature requests.

If possible, it's usually a good idea to add some pointers to the issue author's question.
2. Close the issue and label it with `type/question`.

4. Prioritization of issues

In general bugs and enhancement issues should be labeled with a priority.

This is the most difficult thing with triaging issues since it requires a lot of knowledge, context and experience before being able to think of and start feel comfortable adding a certain priority label.

The key here is asking for help and discuss issues to understand how more experienced project members think and reason. By doing that you learn more and eventually be more and more comfortable with prioritizing issues.

In case there is an uncertainty around the prioritization of an issue, please ask the maintainers for help.

Label	Description
`priority/critical`	Highest priority. Must be actively worked on as someone's top priority right now.
`priority/support`	This is important for one or several customers having a paid Grafana support subscription.
`priority/important-soon`	Must be staffed and worked on either currently, or very soon, ideally in time for the next release.
`priority/important-longterm`	Important over the long term, but may not be staffed and/or may need multiple releases to complete.
`priority/nice-to-have`	Have good idea, but not scheduled for any release.
`priority/awaiting-more-evidence`	Lower priority. Possibly useful, but not yet enough interest in it.
`priority/unscheduled`	Something to look into before and to be discussed during the planning of the next (upcoming) major/minor stable release.

Critical bugs

1. If a bug has been categorized and any of the following criteria apply, the bug should be labeled as critical and must be actively worked on as someone's top priority right now.
   - Results in any data loss
   - Critical security or performance issues
   - Problem that makes a feature unusable
   - Multiple users experience a severe problem affecting their business, users etc.
2. Label the issue `priority/critical`.

3. If applicable, label the issue `priority/support-subscription`.
4. Add the issue to the next upcoming patch release milestone. Create a new milestone if there are none.
5. Escalate the problem to the maintainers.
6. Assign or ask a maintainer for help assigning someone to make this issue their top priority right now.

Important short-term

1. Label the issue `priority/important-soon`.
2. If applicable, label the issue `priority/support-subscription`.
3. Add the issue to the next upcoming patch or major/minor stable release milestone. Ask maintainers for help if unsure if it's a patch or not. Create a new milestone if there are none.
4. Make sure to add the issue to a suitable backlog of a GitHub project and prioritize it or assign someone to work on it now or very soon.
5. Consider requesting help from the community, even though it may be problematic given a short amount of time until it should be released.

Important long-term

1. Label the issue `priority/important-longterm`.
2. Consider requesting help from the community.

Nice to have

1. Label the issue `priority/nice-to-have`.
2. Consider requesting help from the community.

Not critical, but unsure?

1. Label the issue `priority/unscheduled`.
2. Consider requesting help from the community.

5. Requesting help from the community

Depending on the issue and/or priority, it's always a good idea to consider signalling to the community that help from community is appreciated and needed in case an issue is not prioritized to be worked on by maintainers. Use your best judgement. In general, requesting help from the community means that a contribution has a good chance of getting accepted and merged.

In many cases the issue author or community as a whole is more suitable to contribute changes since they're experts in their domain. It's also quite common that someone has tried to get something to work using the documentation without success and made an effort to get it to work and/or reached out to the community site to get the missing information. Particularly in these areas it's more likely that there exist experts in their own domain and it is usually a good idea to request help from contributors:

- Database setups
- Authentication like OAuth providers and LDAP setups
- Platform specific things
- Reverse proxy setups
- Alert notifiers

1. Kindly and politely add a comment to signal to users subscribed to updates of the issue.
   - Explain that the issue would be nice to get resolved, but it isn't prioritized to work on by maintainers for an unforeseen future.
   - If possible or applicable, try to help contributors getting starting by adding pointers and references to what code/files need to be changed and/or ideas of a good way to solve/implement the issue.
2. Label the issue with `help wanted`.
3. If applicable, label the issue with `beginner friendly` to denote that the issue is suitable for a beginner to work on.
4. If possible, try to estimate the amount of work by adding `effort/small`, `effort/medium` or `effort/large`.

Investigation of issues

When an issue has all basic information provided, but the triage responsible haven't been able to reproduce the reported problem at a first glance, the issue is labeled Needs investigation. Depending on the perceived severity and/or number of upvotes, the investigation will either be delegated to another maintainer for further investigation or put on hold until someone else (maintainer or contributor) picks it up and eventually starts investigating it.

Investigating issues can be a very time consuming task, especially for the maintainers, given the huge number of combinations of plugins, data sources, platforms, databases, browsers, tools, hardware, integrations, versions and cloud services, etc that are being used with Grafana. There is a certain number of combinations that are more common than others, and these are in general easier for maintainers to investigate.

For some other combinations it may not be possible at all for a maintainer to setup a proper test environment to investigate the issue. In these cases we really appreciate any help we can get from the community. Otherwise the issue is highly likely to be closed.

Even if you don't have the time or knowledge to investigate an issue we highly recommend that you upvote the issue if you happen to have the same problem. If you have further details that may help investigating the issue please provide as much information as possible.

Automation

We have some automation that triggers on comments or labels being added to issues. Many of these automated behaviors are defined in commands.json. Or in other GitHub Actions

- Add /duplicate `#<issue number>` to have Grafana label & close issue with an appropriate message.
- Add `bot/question` and the bot will close it with an appropriate message.

Read more on bot actions

External PRs

Part of issue triage should also be triaging of external PRs. Main goal should be to make sure PRs from external contributors have an owner/reviewer and are not forgotten.

1. Check new external PRs which do not have a reviewer.
2. Check if there is a link to an existing issue.
3. If not and you know which issue it is solving, add the link yourself, otherwise ask the author to link the issue or create one.
4. Assign a reviewer based on who was handling the linked issue or what code or feature does the PR touches (look at who was the last to make changes there if all else fails).

Appendix

Setting up Gmail filters

If you're using Gmail it's highly recommended that you setup filters to automatically remove email from the inbox and label them accordingly to make it easy for you to understand when you need to act upon a notification or process all incoming issues that haven't been triaged.

This may be setup by personal preference, but here's a working configuration for reference.

1. Follow instructions in gist
2. In Gmail, go to Settings -> Filters and Blocked Addresses
3. Import filters -> select xml file -> Open file
4. Review filters
5. Optional, Check Apply new filters to existing email
6. Create filters

This will give you a structure of labels in the sidebar similar to the following:

```
- Inbox
...
- GitHub (mine)
```

```
  - activity
  - assigned
  - mentions
- GitHub (other)
 - Grafana
```

- All notifications you'll need to read/take action on show up as unread in GitHub (mine) and its sub-labels.
- All other notifications you don't need to take action on show up as unread in GitHub (other) and its sub-labels
  - This is convenient for issue triage and to follow the activity in the Grafana project.