Create a pull request

We're excited that you're considering making a contribution to the Grafana project! This document guides you through the process of creating a pull request.

Before you begin

We know you're excited to create your first pull request. Before we get started, read these resources first:

- Learn how to start Contributing to Grafana.
- Make sure your code follows the relevant style guides.

Your first pull request

If this is your first time contributing to an open-source project on GitHub, make sure you read about Creating a pull request.

To increase the chance of having your pull request accepted, make sure your pull request follows these guidelines:

- Title and description matches the implementation.
- Commits within the pull request follow the Formatting guidelines.
- The pull request closes one related issue.
- The pull request contains necessary tests that verify the intended behavior.
- If your pull request has conflicts, rebase your branch onto the main branch.

If the pull request fixes a bug:

- The pull request description must include `Closes #<issue number>` or `Fixes #<issue number>`.
- To avoid regressions, the pull request should include tests that replicate the fixed bug.

Frontend-specific guidelines

Pull requests for frontend contributions must:

- Use Emotion for styling.
- Not increase the Angular code base.
- Not use `any` or `{}` without reason.
- Not contain large React components that could easily be split into several smaller components.
- Not contain backend calls directly from components—use actions and Redux instead.

Pull requests for Redux contributions must:

- Use the `actionCreatorFactory` and `reducerFactory` helpers instead of traditional switch statement reducers in Redux. Refer to Redux framework for more details.
- Use `reducerTester` to test reducers. Refer to Redux framework for more details.
- Not contain code that mutates state in reducers or thunks.
- Not contain code that accesses the reducers state slice directly. Instead, the code should use state selectors to access state.

Pull requests that add or modify unit tests that are written in Jest must adhere to these guidelines:

- Don't add snapshots tests. We are incrementally removing existing snapshot tests, we don't want more.
- If an existing unit test is written in Enzyme, migrate it to RTL (React Testing Library), unless you're fixing a bug. Bug fixes usually shouldn't include any bigger refactoring, so it's ok to skip migrating the test to RTL.

Pull requests that create new UI components or modify existing ones must adhere to the following accessibility guidelines:

- Use semantic HTML.
- Use ARIA roles, labels and other accessibility attributes correctly. Accessibility attributes should only be used when semantic HTML doesn't satisfy your use case.
- Use the Grafana theme palette for styling. It contains colors with good contrast which aids accessibility.
- Use RTL for writing unit tests. It helps to create accessible components.

Pull requests that introduce accessibility(a11y) errors:

We use pa11y-ci to collect accessibility errors on some URLs on the project, threshold errors are specified per URL.

If the contribution introduces new a11y errors, our continuous integration will fail, preventing you to merge on the main branch. In those cases there are two alternatives for moving forward:

- Check the error log on the pipeline step `test-a11y-frontend-pr`, identify what was the error, and fix it.
- Locally run the command `yarn test:accessibility-report` that generates an HTML accessibility report, then go to the URL that contains your change, identify the error, and fix it. Keep in mind, a local e2e Grafana instance is going to be running on `http://localhost:3001`.

You can also prevent introducing a11y errors by installing an a11y plugin in your browser, for example, axe DevTools, Accessibility Insights for Web among others.

Backend-specific guidelines

Please refer to the backend style guidelines.

Code review

Once you've created a pull request, the next step is to have someone review your change. A review is a learning opportunity for both the reviewer and the author of the pull request.

If you think a specific person needs to review your pull request, then you can tag them in the description or in a comment. Tag a user by typing the `@` symbol followed by their GitHub username.

We recommend that you read How to do a code review to learn more about code reviews.

Formatting guidelines

A well-written pull request minimizes the time to get your change accepted. These guidelines help you write good commit messages and descriptions for your pull requests.

Commit message format

Grafana uses the guidelines for commit messages outlined in How to Write a Git Commit Message, with the following additions:

- Subject line must begin with the *area* of the commit.
- A footer in the form of an optional keyword and issue reference.

Area

The area should use upper camel case, e.g. UpperCamelCase.

Prefer using one of the following areas:

- **Build:** Change the build system, or external dependencies
- **Chore:** Change that don't affect functionality
- **Dashboard:** Change the Dashboard feature
- **Docs:** Change documentation
- **Explore:** Change the Explore feature
- **Plugins:** Change the ... plugin

For changes to data sources, the area is the name of the data source. For example, AzureMonitor, Graphite, or Prometheus.

For changes to panels, the area is the name of the panel, suffixed with Panel. For example, GraphPanel, SinglestatPanel, or TablePanel.

Examples

- `Build: Support publishing MSI to grafana.com`
- `Explore: Add Live option for supported data sources`
- `GraphPanel: Fix legend sorting issues`
- `Docs: Change url to URL in all documentation files`

If you're unsure, see the existing changelog for inspiration or guidance.

Pull request titles

The Grafana team *squashes* all commits into one when we accept a pull request. The title of the pull request becomes the subject line of the squashed commit message. We still encourage contributors to write informative commit messages, as they becomes a part of the Git commit body.

We use the pull request title when we generate change logs for releases. As such, we strive to make the title as informative as possible.

Make sure that the title for your pull request uses the same format as the subject line in the commit message.

Configuration changes

If your PR includes configuration changes, all of the following files must be changed correspondingly:

- conf/defaults.ini
- conf/sample.ini
- docs/sources/administration/configuration.md