Grafana workflow

This document is based on GOVERNANCE.md. We assume good faith and intend to keep all processes as lightweight as possible but as specific as required. In case of disagreements about anything in this document, GOVERNANCE.md applies.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119.

Git and GitHub terminology are used throughout this document.

Team members and their access to repositories is maintained through GitHub teams. Team maintainers add and remove team members as outlined in GOVERNANCE.md.

Code changes

Proposing changes

Examples of proposed changes are overarching architecture, component design, and specific code or graphical elements. Proposed changes SHOULD cover the big picture and intention, but individual parts SHOULD be split into the smallest possible changes. Changes SHOULD be based on and target the main branch. Depending on size of the proposed change, each change SHOULD be discussed, in increasing order of change size and complexity:

- Directly in a RR (Pull Request) - this MAY be done, but SHOULD not be the common case.
- Issue
- Developer mailing list
- Design document, shared via Google Docs, accessible to at least all team members.

Significant changes MUST be discussed and agreed upon with the relevant subsystem maintainers.

Merging PRs (Pull Requests)

Depending on the size and complexity of a PR, different requirements MUST be applied. Any team member contributing substantially to a PR MUST NOT count against review requirements. Commits MUST be merged into main using PRs. They MUST NOT be merged into main directly.

- Every merge MUST be approved by at least one team member.
- Non-trivial changes MUST be approved by at least
  - two team members, or
  - one subsystem maintainer.

- Significant changes MUST be approved by at least
  - two team members, AND
  - the relevant subsystem maintainer.

PRs MUST be reviewed and approved via GitHub's review system.

- Reviewers MAY write comments if approving
- Reviewers MUST write comments if rejecting a PR or if requesting changes.

Once a PR is approved as per above, any team member MAY merge the PR.

Backporting a PR

PRs intended for inclusion in the next PATCH release they must be backported to the release branch. The bot can do this automatically. Read more on backport PRs. Both the source PR and the backport PR should be assigned to the patch release milestone, unless you are backporting to many releases then it can differ.

Backport PRs are also needed during the beta period to get fixes into the stable release.

Release workflow

Branch structure

Grafana uses trunk-based development.

In particular, we found that the following principles match how we work:

- Main and release branches MUST always build without failure.
- Branches SHOULD be merged often. Larger changes SHOULD be activated with feature flags until they are ready. Long-lived development branches SHOULD be avoided.
- Changes MAY be enabled by default once they are in a complete state
- Changes which span multiple PRs MUST be described in an overarching issue or Google Doc.

Releases

Releases MUST follow Semantic Versioning in naming and SHOULD follow Semantic Versioning as closely as reasonably possible for non-library software.

Release branches MUST be split from the following branches.

- MAJOR release branches MUST be based on main.
- MINOR release branches MUST be based on main.
- PATCH release branches MUST be split from the relevant MINOR release branch's most current PATCH

Security releases follow the same process but MUST be prepared in secret. Security releases MUST NOT include changes which are not related to the security fix. Normal release processes MUST accommodate the security release process. SECURITY.md MUST be followed.

Releases follow the following cadence

- MAJOR: Yearly
- MINOR: Every 4-6 weeks
- PATCH: As needed

Releases SHOULD NOT be delayed by pending changes.

Releases MUST be coordinated with the relevant subsystem maintainers.