Testing sandboxes with Git Sync

- Use it for
- Architecture
- Repository structure
- Configuration parameters
- How it works
- Individual or shared sandboxes
- Protect sandbox branches
- Promote a sandbox to main
  - Option 1: Open a pull request from the sandbox branch (recommended)
  - Option 2: Cherry-pick or merge selected changes
  - Option 3: Promote through a separate instance or path
  - Option 4: Promote from the Grafana UI
- Learn more

Use a sandbox branch that users can push to directly, without opening a pull request for every change. A sandbox branch is a low-friction space for short-lived projects and experimentation, while keeping every dashboard version-controlled in Git. When work is ready, you promote it to the `main` branch.

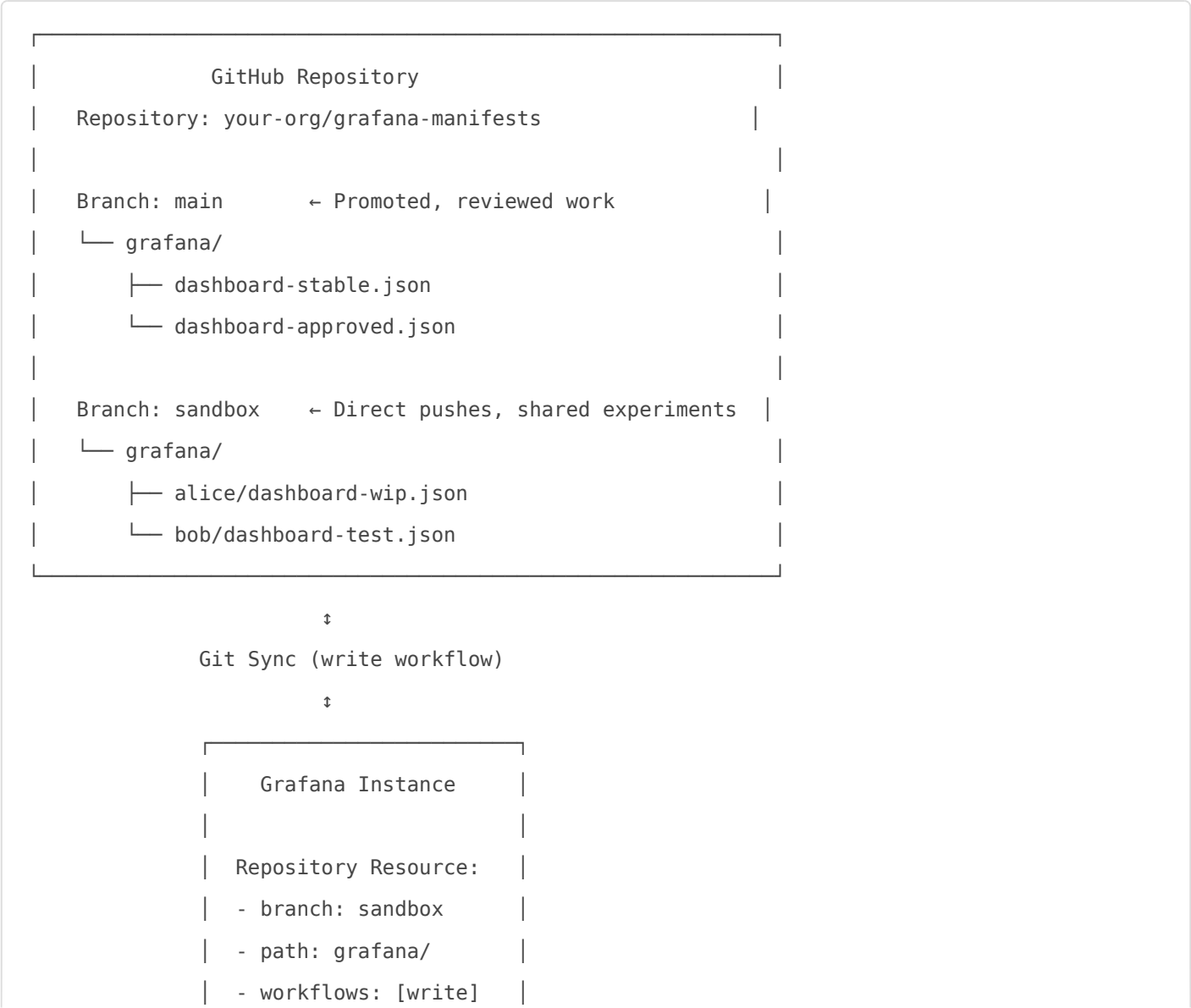
A sandbox is a branch in the same repository, synced to Grafana with the `write` workflow enabled. With the `write` workflow, saving a dashboard commits directly to the sandbox branch, so the editing experience feels close to using Grafana without Git Sync. The difference is that every change is still committed to Git, so nothing is lost and history is preserved.

This scenario uses a single **shared** sandbox branch that everyone works in. A shared sandbox uses only one Repository connection regardless of how many users work in it, which keeps you well within the per-stack limit of 10 repository connections. If you need stricter isolation, you can instead give each user their own sandbox branch. Refer to Individual or shared sandboxes.

Use it for

- **Short-lived projects**: You need a temporary space for work that may or may not reach production.
- **Shared experimentation**: A team experiments freely in one space without affecting shared production dashboards.
- **Fast iteration**: You want the low-friction editing of a non-provisioned instance, but with the safety of Git history.
- **Staged promotion**: You develop in a sandbox, then promote finished work to the `main` branch.

Architecture



```
                │                          │
                │  Save = direct commit    │
                │  to sandbox              │
                └──────────────────────────┘
```

Repository structure

In Git:

```
your-org/grafana-manifests
├── (branch: main)
│   └── grafana/
│       ├── dashboard-stable.json
│       └── dashboard-approved.json
└── (branch: sandbox)
    └── grafana/
        ├── alice/
        │   └── dashboard-wip.json
        └── bob/
            └── dashboard-test.json
```

In Grafana Dashboards view (synced to `sandbox`):

```
Dashboards
└── □□grafana-manifests/
    ├── □□alice/
    │   └── Work in Progress Dashboard
    └── □□bob/
        └── Test Dashboard
```

- A folder named "grafana-manifests" (from the repository name) contains the dashboards synced from the sandbox branch.
- Users can create subfolders inside it, such as `alice/` and `bob/`, to keep their work separate within the shared branch. Subfolders map to subdirectories under the synced path, so no extra Repository connection is needed.
- Only dashboards on the configured branch and path appear in the instance.
- Saving a dashboard commits directly to the sandbox branch.

Configuration parameters

Configure the Repository resource to sync with the sandbox branch and enable the `write` workflow:

- **Repository**: `your-org/grafana-manifests`
- **Branch**: `sandbox`
- **Path**: `grafana/`
- **Workflows**: `write`

In the Repository resource, this looks like:

```
spec:
  github:
    url: 'https://github.com/your-org/grafana-manifests'
    branch: 'sandbox'
    path: grafana/
  workflows:
    - write
```

{{< admonition type="note" >}}

If you enable only the `write` workflow, saving a dashboard always commits directly to the sandbox branch. To also allow users to open pull requests from Grafana, add the `branch` workflow: `workflows: [write, branch]`. Refer to Configuration parameters for details.

{{< /admonition >}}

How it works

1. Create a sandbox branch in your repository, for example `sandbox`.
2. Configure a Repository resource that syncs that branch with the `write` workflow enabled.
3. Users create and edit dashboards in Grafana, optionally in their own subfolder. Each save commits directly to the sandbox branch.
4. The experience feels similar to using Grafana without Git Sync, but every change is committed to Git.
5. When work is ready, you promote it to the `main` branch. Refer to Promote a sandbox to main.

Individual or shared sandboxes

A sandbox can be shared by everyone or scoped to a single user. Each sandbox is a branch synced through its own Repository connection, so the choice is shaped by the per-stack limit of 10 repository connections. Refer to Usage and performance limitations for details.

- **Shared sandbox (recommended)**: Use a single branch, such as `sandbox`, synced through one Repository connection that all users share. This uses only one connection regardless of how many users work in it, so it scales to any number of users. Separate users' work with subfolders (for example, `alice/`, `bob/`) if you want some organization within the shared branch. The trade-off is less isolation, since everyone commits to the same branch. This is the setup described above.
- **Individual sandboxes**: Give each user their own branch, such as `sandbox/alice` and `sandbox/bob`, each synced through a separate Repository connection. This provides the most isolation, but every sandbox consumes one of your 10 connections, so per-user sandboxes don't scale to many users. Reserve them for a small number of active users or short-lived projects, and delete the branch and its connection when the work is done to free the connection.

Both approaches promote to `main` the same way. Refer to Promote a sandbox to main.

Protect sandbox branches

Even though sandbox branches allow direct pushes, protect them against destructive operations. We recommend blocking force pushes on sandbox branches in your Git provider.

Force pushes rewrite branch history and can silently discard commits that Git Sync relies on. Blocking them keeps history append-only, so Grafana and Git stay consistent and you can always trace how a dashboard reached its current state.

- **GitHub**: Add a branch protection rule (or ruleset) for the sandbox branch (for example, `sandbox` or the pattern `sandbox/*` if you use individual sandboxes) and disable **Allow force pushes**.
- **GitLab**: Set the sandbox branches as protected and disallow force push.
- **Bitbucket**: Add a branch permission for the sandbox branch that prevents rewriting history.

You typically keep pull requests optional on sandbox branches so users can push directly, while still blocking force pushes.

Promote a sandbox to main

When sandbox work is ready, promote it to the `main` branch. Because everything already lives in Git, promotion is a standard Git operation. Choose the option that fits your review requirements:

Option 1: Open a pull request from the sandbox branch (recommended)

Use your Git provider to open a pull request from the sandbox branch into `main`.

1. In your Git provider, open a pull request from `sandbox` into `main`.
2. Review the changes as a normal Git diff and request approvals as needed.
3. Merge the pull request. The instance syncing `main` picks up the promoted dashboards on its next sync.

This keeps a full review and audit trail for everything that reaches `main`, and is the recommended path when `main` feeds production.

If you also enable the `branch` workflow on the sandbox Repository resource, users can open these pull requests directly from Grafana when they save, instead of switching to the Git provider.

Option 2: Cherry-pick or merge selected changes

When only some dashboards are ready, promote them selectively instead of merging the whole branch.

1. Cherry-pick the relevant commits from the sandbox branch onto a branch based on `main`, or copy the specific dashboard files.
2. Open a pull request into `main` and review it.
3. Merge to complete the promotion.

Use this when a shared sandbox contains a mix of finished and experimental work.

Option 3: Promote through a separate instance or path

When `main` is synced to a production instance, treat the sandbox as the development stage of a promotion flow.

1. Develop in the sandbox instance (synced to the sandbox branch with the `write` workflow).
2. Promote approved dashboards to `main` using a pull request (Option 1) or by copying files into the `main` path.
3. The production instance syncing `main` updates automatically.

This mirrors the development and production environments scenario, with the sandbox branch acting as the development stage.

Option 4: Promote from the Grafana UI

If the `main` branch is synced to a folder in the same instance (through a second Repository connection) or to a separate instance, users can move a dashboard into it directly from Grafana, without touching Git:

- **Save a copy**: Open the sandbox dashboard, select **Save as** (save a copy), and choose the folder backed by the `main` branch as the destination. Git Sync commits the copy to `main`.
- **Import dashboard**: Export the dashboard from the sandbox (**Export** as JSON), then use **Dashboards > New > Import** and save it into the folder backed by the `main` branch. Git Sync commits it to `main`.

This suits users who prefer working entirely in the UI. Note that these paths commit to `main` without a pull request, so use them only when `main` allows direct writes; when `main` requires review, prefer Option 1.

After promotion, you can delete short-lived sandbox branches, or keep a long-lived shared sandbox and continue iterating.

Learn more

Refer to the following documents to learn more:

- Git Sync for development and production environments
- Git Sync key concepts
- Configure Git Sync as code
- Manage provisioned dashboards
- Usage and performance limitations

□□□ #2
□ □□□ □□□ 2026-10-31 15:50:10 UTC
□ □□□ □□□ 2026-10-31 15:50:13 UTC