Git Sync key concepts

- Key Git Sync components
  - Grafana instance
  - Git repository
    - Flexible configuration for your Git Sync repositories
  - Git Sync repository resource
  - Git Sync connection resource
- How does Git Sync behave?
  - Sync targets
    - Multiple folderless repositories
  - Git Sync states
- Example: Relationship between repository, branch, and path

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

Git Sync functionalities are constantly evolving. Contact Grafana for support or to report any issues you encounter and help us improve this feature.

{{< /admonition >}}

Key Git Sync components

Before you start using Git Sync, understand how the key Git Sync components relate:

- Grafana instance
- Git repository
- Git Sync repository resource
- Git Sync connection resource

Grafana instance

A Grafana instance is a running Grafana server. Multiple instances can:

- Connect to the same Git repository using different Repository configurations.
- Sync from different branches of the same repository.
- Sync from different paths within the same repository.
- Sync from different repositories.

Git repository

A Git repository is the external storage you want to sync your Grafana instance with. You can organize your Git repository in several ways:

- Single branch, multiple paths: Use different directories for different purposes. For example, `dev/`, `prod/`, or `team-a/`.
- Multiple branches: Use different branches for different environments or teams. For example, `main`, `develop`, or `team-a`.
- Multiple repositories: Use separate repositories for different teams or environments.

Flexible configuration for your Git Sync repositories

Git Sync repositories support different combinations of repository URL, branch, and path:

- Different Git repositories: Each environment or team can use its own repository.
  - Instance A: `repository: your-org/grafana-prod`.
  - Instance B: `repository: your-org/grafana-dev`.
- Different branches: Use separate branches within the same repository.
  - Instance A: `repository: your-org/grafana-manifests, branch: main`.
  - Instance B: `repository: your-org/grafana-manifests, branch: develop`.
- Different paths: Use different directory paths within the same repository.
  - Instance A: `repository: your-org/grafana-manifests, branch: main, path: production/`.
  - Instance B: `repository: your-org/grafana-manifests, branch: main, path: development/`.
- Any combination: Mix and match based on your workflow requirements.

Git Sync repository resource

A repository resource is a Grafana configuration object that defines the connection between a group of repositories and your Grafana instance via Git Sync.

- Which Git repository to sync with your Grafana instance.
- Which branch to use.
- Which directory path to synchronize.
- Sync behavior and workflows.

Each repository resource creates bidirectional synchronization between a Grafana instance and a specific location in Git.

Git Sync connection resource

A connection is the authentication setup between Grafana and an external Git provider, required when you're not using a Personal Access Token or a static token to authenticate. Use it to authorize access to your external repositories, and to generate or refresh the credentials for Git Sync. A single connection can be reused across multiple repositories.

For example, if you're using GitHub App to authenticate, the connection represents the app installation. Grafana uses this connection to authenticate with GitHub, create access tokens, and authorize repository access on your behalf.

The connection resource includes:

- The external provider configuration: The authentication mechanism Grafana uses to communicate with the Git provider (for example, a GitHub App).
- The provider authorization or installation: The authorized entity that allows Grafana to request tokens from the provider.
- Repository access scope: The set of repositories that the connection (and therefore, Grafana) is authorized to access in the Git provider.

How does Git Sync behave?

Git Sync is bidirectional, and syncs a repository resource with your Grafana instance. You can modify provisioned resources both from the Grafana UI or from the synced GitHub repository, and changes will be reflected in both places:

1. Grafana monitors the specified Git location (repository, branch, and path).
2. Grafana creates a folder in Dashboards (typically named after the repository).
3. Grafana creates dashboards from dashboard JSON files in Git within this folder.
4. Grafana commits dashboard changes made in the UI back to Git.
5. Grafana pulls dashboard changes made in Git and updates dashboards in the UI.
6. Synchronization occurs at regular intervals (configurable), or instantly if you use webhooks.

You can find the provisioned dashboards organized in folders under **Dashboards**.

Sync targets

With Git Sync you can place synced resources in Grafana in two ways:

- **Folder sync**: Grafana creates a folder named after the repository and places all synced resources inside it. Subdirectories in the repository become subfolders within that folder. **This is the default behavior**.
- **Folderless sync**: Grafana places synced resources at the top level, without creating a wrapper folder. Files at the repository path root become top-level resources, and subdirectories become top-level folders.

Use folder sync to keep each repository's resources grouped together under a dedicated folder. Use folderless sync when you want provisioned resources to appear at the top of your Dashboards view instead of nested inside a repository folder.

Both modes can coexist with each other and with resources that aren't managed by Git Sync.

The following examples use the same repository to show how the same files appear with each mode.

The repository `grafana-manifests` syncs from the path `grafana/`:

```
your-org/grafana-manifests/
└── grafana/
    ├── cpu-metrics.json
    └── team-platform/
        ├── _folder.json
        └── memory-usage.json
```

The `_folder.json` file stores the `team-platform` folder's stable UID and display name, so the folder keeps its identity and permissions if you move or rename it in the repository. Refer to The Git Sync folder metadata file for details.

The instance also has content that isn't managed by Git Sync: a manually created **Ops** folder and an **Ad-hoc dashboard**.

With folder sync, a repository folder wraps the synced resources, alongside the unprovisioned content:

```
Dashboards
├── □□grafana-manifests/      ← managed by Git Sync
│   ├── CPU Metrics Dashboard
│   └── □□team-platform/
│       └── Memory Usage Dashboard
├── □□Ops/                    ← not managed by Git Sync
│   └── Ops dashboard
└── Ad-hoc dashboard           ← not managed by Git Sync
```

With folderless sync, the same files map to the top level, next to the unprovisioned content:

```
Dashboards
├── CPU Metrics Dashboard      ← managed by Git Sync
├── □□team-platform/          ← managed by Git Sync
│   └── Memory Usage Dashboard
├── □□Ops/                    ← not managed by Git Sync
│   └── Ops dashboard
└── Ad-hoc dashboard           ← not managed by Git Sync
```

Folderless sync only manages the resources it provisions. The unprovisioned **Ops** folder and **Ad-hoc dashboard** are left untouched.

Multiple folderless repositories

Because folderless sync doesn't create a wrapper folder, several folderless repositories can sync to the top level at the same time. Each repository manages only the resources it provisions:

```
Dashboards
├── CPU Metrics Dashboard      ← managed by grafana-manifests
├── □□team-platform/          ← managed by grafana-manifests
│   └── Memory Usage Dashboard
├── Billing Overview           ← managed by finance-dashboards
├── □□invoices/               ← managed by finance-dashboards
│   └── Monthly Invoices
└── Ad-hoc dashboard           ← not managed by Git Sync
```

Git Sync states

Your Grafana instance can be in one of the following Git Sync states:

- **Unprovisioned**: None of the instance's resources are being managed by Git Sync.
- **Partially provisioned**: Some of the resources are controlled by Git Sync.
- **Fully provisioned**: All supported resource types are managed by Git Sync. Unsupported resources are **not managed**.

Example: Relationship between repository, branch, and path

Here's an example showing how the repository, branch, and path concepts work together:

Configuration:

- **Repository**: `your-org/grafana-manifests`
- **Branch**: `main`
- **Path**: `team-platform/grafana/`

In Git (on branch `main`):

```
your-org/grafana-manifests/
├── .git/
├── README.md
├── team-platform/
│   └── grafana/
│       ├── cpu-metrics.json       ← Synced
│       ├── memory-usage.json      ← Synced
│       └── disk-io.json           ← Synced
├── team-data/
│   └── grafana/
│       └── pipeline-stats.json    ← Not synced (different path)
└── other-files.txt                ← Not synced (outside path)
```

In the Grafana Dashboards view:

```
Dashboards
└── □□grafana-manifests/
    ├── CPU Metrics Dashboard
    ├── Memory Usage Dashboard
    └── Disk I/O Dashboard
```

Key takeaways:

- Grafana only synchronizes files within the specified path (`team-platform/grafana/`).
- Grafana ignores files in other paths or at the repository root.
- The folder name in Grafana comes from the repository name.
- Dashboard titles come from the JSON file content, not the filename.

□□□ #2
□ □□□ □□□ 2026-10-31 15:49:04 UTC
□ □□□ □□□ 2026-10-31 15:49:06 UTC