keepit

Keepit for GitHub

Keepit for **1**

Coverage + Limitations **1**

Limitations **2**

Branches *2*

Pull Requests *2*

Pull Request participants *3*

Pull Request metadata *3*

Pull Request target/source branch *3*

Repository content *3*

Repository settings *4*

Repository participants *4*

Users *4*

Teams *4*

User's email *5*

Coverage + Limitations

Introducing code-level protection for Git repositories including all branches, tags, commits, and history. This foundational layer ensures code is safe and fully restorable, with metadata backup and compliance tooling planned for future iterations.

Repository

- **Commits**
- **Branches**
- **Pull Requests**
  - **Pull Request participants**
  - **Pull Request metadata**
  - **Pull Request target/source branch**
  - **Pull Request commits**
- **Repository Content**

Users

Teams

Limitations

Branches

Branches that are stored directly under the Repository folder are restorable, but only if the entire organization was not reset. A branch can be successfully restored only if the commits referenced by this branch were not deleted from the repository. If all repositories and their full commit history were deleted, branches can be restored only through the Git Native Protocol.

Please note: If a branch was renamed on the remote without being deleted, and you choose to restore it using its old name from a previous snapshot, both versions will coexist. This results in two identical branches with different names.

Pull Requests

Restore functionality for Pull Requests is not implemented yet. At the moment, both open and closed pull requests are backed up together with pull request participants (author, reviewers, assignees, and teams), pull request metadata (source and target branches and pull request description). If the source or target branch exists, a redirect is created to the corresponding branch folder.

Closed and open pull requests may have the same source and target branches. When creating a pull request using GitHub REST API endpoints, a pull request can only be created with the status open, and GitHub does not allow multiple open pull requests with the same head (source) and base (target) branches at the same time. This limitation prevents full recreation of all backed-up pull requests during restore.

Pull Request participants

The Pull Request Participants file archives the users and teams associated with the pull request, specifically the author, assignees, requested reviewers, and requested review teams. This file follows a sparse data model: fields are only populated if the corresponding data exists on GitHub (e.g., if no reviewers are assigned, that field is omitted). Currently, this file serves as an informational backup record and **does not support restoration.**

Pull Request metadata

The Pull Request Metadata file captures the essential context of the pull request, including its description and the specific source (head) and target (base) branch names. Crucially, this file persists the branch names even if the source or target branches are subsequently deleted from the repository. While this file is a mandatory component of every snapshot, it is currently intended for data preservation only and is **not restorable.**

Pull Request target/source branch

The Source and Target branch folders function as references to the corresponding branches located in the root Branches directory. These folders are conditional: they are generated only if the respective branch currently exists on the remote GitHub repository. If a branch has been deleted and the folder is absent, the branch details can still be verified in the Metadata file located at the same directory level. These folders serve as organizational references and are **not restorable entities.**

Repository content

The downloaded archive contains a standard **.git** directory structure, accompanied by execution scripts for **Windows** (.cmd) and **Linux** (.sh). These scripts are provided for customer's convenience to automate the restoration process and are described in detail on the GitHub Support pages.

The included **.git** folder mirrors the standard structure of a local Git repository. This compatibility ensures that the data can be interpreted by native Git tools without modification.

The critical components within this directory include:

- **HEAD (File):** A reference pointer that determines the currently checked-out branch. In the context of a backup, this typically points to the repository's default branch (e.g., refs/heads/main or master) as it existed on the remote at the time of the snapshot.
- **objects (Folder):** The core object database of the repository. This directory contains the actual content of your files, commits, and directory trees, stored efficiently as compressed **packfiles**. This is the heavy lifting of the backup; without this folder, the file history does not exist.
- **refs (Folder):** A directory structure that stores the "bookmarks" of the repository. It contains text files holding the specific commit hashes (SHA-1) that correspond to:
    - **heads:** Local branches (e.g., main, feature/login).
    - **tags:** Release tags (e.g., v1.0, v2.0).

Repository settings

The Repository Settings file captures the configuration parameters found in the "Settings" tab of the remote GitHub repository. This includes metadata such as repository visibility (Public/Private), the default branch name, homepage, and merge strategies (e.g., "Allow Auto-merge"). These settings are currently archived **for reference only** and cannot be restored.

Repository participants

The Repository Participants file records the individuals and groups associated with the repository, specifically direct collaborators, contributors, and assigned teams. Adhering to a sparse data model, fields are only populated when corresponding data exists on GitHub (e.g., if no teams are assigned, the field is omitted). This file serves as an informational backup record and **does not currently support restoration**.

- When a repository has a very large number of contributors, the GitHub API may refuse to return the full list. This typically occurs once a repository reaches around 5,000 contributors or in exceptionally large repositories.
- Since the complete list cannot be retrieved in these cases, the following message will be added to the file content instead:
- The contributor list is too large and could not be retrieved for preview. All contributor data is included in the backup.
- GitHub calculates the number of contributors based on user accounts rather than individual email addresses. This means that if a single user has committed using multiple email addresses, all of those commits are attributed to the same GitHub account and counted as one contributor.
- Additionally, if only the main branch is forked or mirrored, contributions made to other branches in the original repository are not included.
- Due to these factors, **the number of contributors displayed in our system may be lower than the number shown in the original repository instance**.

Users

The Users folder catalogs all members and collaborators associated with the repository. Within each user's specific subfolder, a Definition file contains key metadata such as email, user ID, and

membership type. User accounts **cannot be restored**, as the GitHub REST API does not provide endpoints to programmatically create or recreate user profiles.

Teams

The Teams folder lists all teams assigned to the repository, mirroring GitHub's support for nested team hierarchies (parent and child teams). Each specific team folder contains two primary elements:

- Definition File: Stores team attributes such as Name, Description, and Privacy settings.
- Members Folder: Contains references that link back to the specific user profiles stored in the global **Users** directory.

The entire Teams directory structure is for backup purposes only and **is not restorable**.

User's email

To ensure a user's email address is backed up, it must be explicitly added to their User Profile.