Using the REST API to interact with your Git database

In this article

Use the REST API to read and write raw Git objects to your Git database on GitHub Enterprise Server and to list and update your references (branch heads and tags).

Overview

This basically allows you to reimplement a lot of Git functionality with the REST API - by creating raw objects directly into the database and updating branch references you could technically do just about anything that Git can do without having Git installed.

The REST API will return a `409 Conflict` if the Git repository is empty or unavailable. An unavailable repository typically means GitHub Enterprise Server is in the process of creating the repository. For an empty repository, you can use the "Repositories" endpoint to create content and initialize the repository so you can use the API to manage the Git database. Contact your site administrator if this response status persists.

For more information on the Git object database, please read the Git Internals chapter of the Pro Git book.

As an example, if you wanted to commit a change to a file in your repository, you would:

- Get the current commit object
- Retrieve the tree it points to
- Retrieve the content of the blob object that tree has for that particular file path
- Change the content somehow and post a new blob object with that new content, getting a blob SHA back
- Post a new tree object with that file path pointer replaced with your new blob SHA getting a tree SHA back
- Create a new commit object with the current commit SHA as the parent and the new tree SHA, getting a commit SHA back
- Update the reference of your branch to point to the new commit SHA

It might seem complex, but it's actually pretty simple when you understand the model and it opens up a ton of things you could potentially do with the API.

Checking mergeability of pull requests

Warning! Please do not depend on using Git directly or `GET /repos/{owner}/{repo}/git/refs/{ref}` for updates to `merge` Git refs, because this content becomes outdated without warning.

A consuming API needs to explicitly request a pull request to create a *test* merge commit. A *test* merge commit is created when you view the pull request in the UI and the "Merge" button is displayed, or when you get, create, or edit a pull request using the REST API. Without this request, the `merge` Git refs will fall out of date until the next time someone views the pull request.

If you are currently using polling methods that produce outdated `merge` Git refs, then GitHub recommends using the following steps to get the latest changes from the default branch:

1. Receive the pull request webhook.
2. Call `GET /repos/{owner}/{repo}/pulls/{pull_number}` to start a background job for creating the merge commit candidate.
3. Poll your repository using `GET /repos/{owner}/{repo}/pulls/{pull_number}` to see if the `mergeable` attribute is `true` or `false`. You can use Git directly or `GET /repos/{owner}/{repo}/git/refs/{ref}` for updates to `merge` Git refs only after performing the previous steps.