Working with large files

While Git is terrific for a great number of use cases, it has trouble with large files. If you are pushing large files to GitHub, you might want to evaluate your workflow to make sure those files are truly necessary. Game assets, such as graphics, might be required for your repository, while SQL database dumps probably aren't.

GitHub warns you when you try to add a file larger than 50 MB. Pushes containing files larger than 100 MB are rejected for a few reasons.

In many cases, committing large files is unintentional and causes unneeded repository bloat. Every time someone clones a repository with a large file, they'll have to fetch that file, adding excess time to their download.

In addition, if a repository is 10 GB in size, Git's architecture requires *another* 10 GB of extra free space available at all times. This allows Git to move the files around in its normal course of operations. Unfortunately, this also means that we must be much less flexible with how we store these repositories.

Article versions

GitHub.com
GitHub Enterprise 2.0
GitHub Enterprise 11.10.340

Warnings and errors on push

When pushing to GitHub, you'll receive a warning or error message if you either add a new file *or* update an existing file that is larger than 50 MB.

The messages on push tell you which files are causing problems:

```
remote: warning: Large files detected.
remote: warning: File big_file is 55.00 MB; this is larger than GitHub's recommended
maximum file size of 50 MB
```

The push with `big_file` is received and saved into the repository on GitHub, but you should consider removing the file and the commit entirely.

```
remote: warning: Large files detected.
remote: error: File giant_file is 123.00 MB; this exceeds GitHub's file size limit of
100 MB
```

This push was rejected because of `giant_file`. The commits pushed are not saved into the repository on GitHub.

Fixing the problem

Warning: This will remove the file entirely from your computer and GitHub. You should make a backup copy if the file is important.

To fix the problem, you'll want to completely remove the large file from Git. If the file was added with your most recent commit, you can just delete the file and amend the commit.

Enter the following in your command line:

```
$ git rm --cached giant_file
# Stage our giant file for removal, but leave it on disk
```

```
$ git commit --amend -CHEAD
# Amend the previous commit with your change
# Simply making a new commit won't work, as you need
# to remove the file from the unpushed history as well

$ git push
# Push our rewritten, smaller commit
```

You may need to remove large files from *far back* in your repository history; the quickest way to do this is with The BFG (a faster, simpler alternative to `git-filter-branch`):

```
$ bfg --strip-blobs-bigger-than 50M
# Git history will be cleaned - files in your latest commit will *not* be touched
```

See The BFG's documentation for full usage and download instructions.

Excessively large files that have already been pushed to your GitHub repositories. However, if you update any of these files locally, you won't be able to push the update.

Alternatives

Our article on managing your repository's size has some helpful information on working with large media files, as well as a list of suggested places to store your files online.

Contact a human

Terms of Service Privacy Security Support