Support for Subversion clients

In this article

GitHub repositories can be accessed from both Git and Subversion (SVN) clients. This article covers using a Subversion client on GitHub and some common problems that you might run into.

GitHub supports Subversion clients via the HTTPS protocol. We use a Subversion bridge to communicate svn commands to GitHub.

Note: Subversion support will be removed from GitHub on January 8, 2024. A future release of GitHub Enterprise Server after January 8, 2024 will also remove Subversion support. To read more about this, see the GitHub blog.

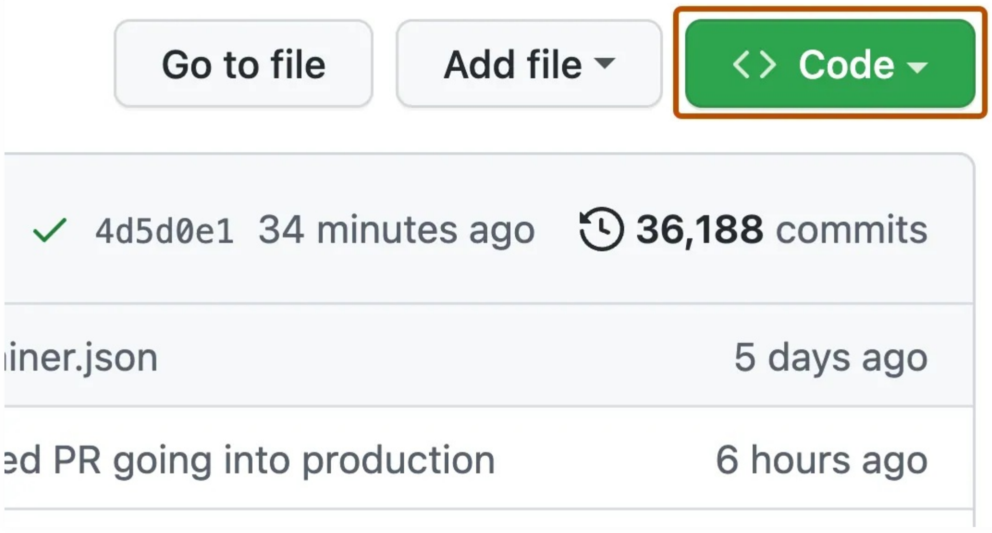
Supported Subversion features on GitHub

Checkout

The first thing you'll want to do is a Subversion checkout. Since Git clones keep the working directory (where you edit files) separate from the repository data, there is only one branch in the working directory at a time.

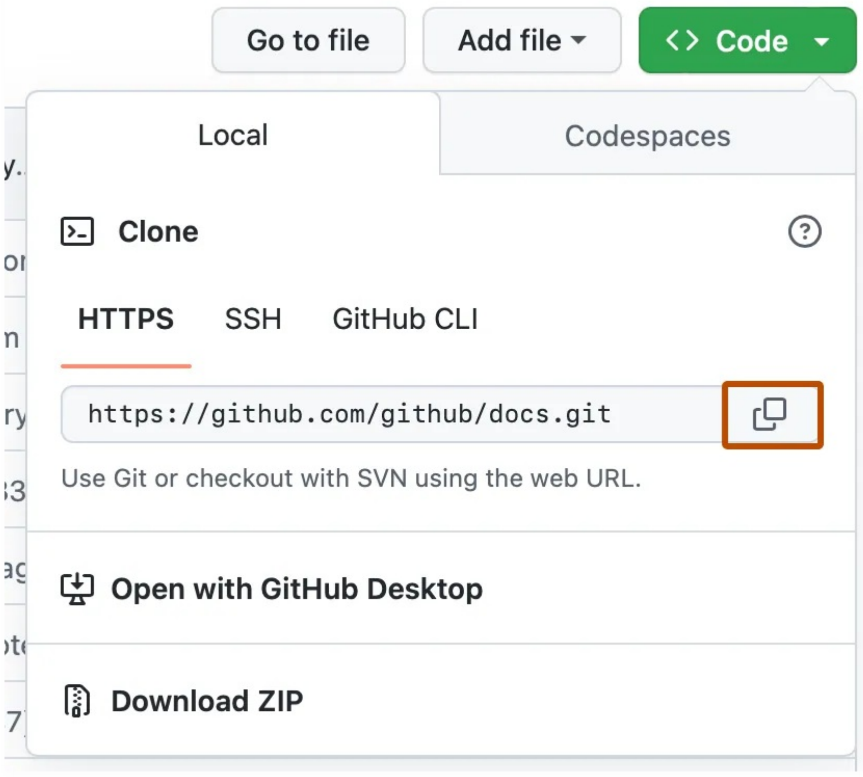
Subversion checkouts are different: they mix the repository data in the working directories, so there is a working directory for each branch and tag you've checked out. For repositories with many branches and tags, checking out everything can be a bandwidth burden, so you should start with a partial checkout.

1. On your GitHub Enterprise Server instance, navigate to the main page of the repository.
2. Above the list of files, click <> **Code**.



3. Copy the URL for the repository.

   - To clone the repository using HTTPS, under "HTTPS", click .
   - To clone the repository using an SSH key, including a certificate issued by your organization's SSH certificate authority, click **SSH**, then click .
   - To clone a repository using GitHub CLI, click **GitHub CLI**, then click .



4. Make an empty checkout of the repository:

```
$ svn co --depth empty https://github.com/USER/REPO
> Checked out revision 1.
$ cd REPO
```

5. Get the `trunk` branch. The Subversion bridge maps trunk to the Git HEAD branch.

```
$ svn up trunk
> A    trunk
> A    trunk/README.md
> A    trunk/gizmo.rb
> Updated to revision 1.
```

6. Get an empty checkout of the `branches` directory. This is where all of the non-`HEAD` branches live, and where you'll be making feature branches.

   ```
   $ svn up --depth empty branches
   Updated to revision 1.
   ```

Creating branches

You can also create branches using the Subversion bridge to GitHub.

From your svn client, make sure the default branch is current by updating `trunk`:

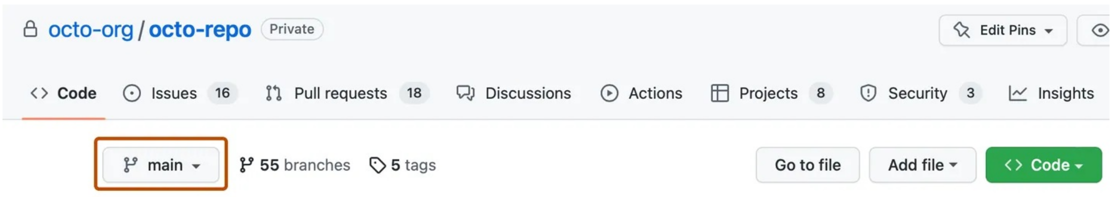
```
$ svn up trunk
> At revision 1.
```

Next, you can use `svn copy` to create a new branch:

```
$ svn copy trunk branches/more_awesome
> A    branches/more_awesome
$ svn commit -m 'Added more_awesome topic branch'
> Adding    branches/more_awesome

> Committed revision 2.
```

You can confirm that the new branch exists in the repository's branch dropdown:



You can also confirm the new branch via the command line:

```
$ git fetch
> From https://github.com/USER/REPO/
> * [new branch]    more_awesome -> origin/more_awesome
```

Making commits to Subversion

After you've added some features and fixed some bugs, you'll want to commit those changes to GitHub. This works just like the Subversion you're used to. Edit your files, and use `svn commit` to record your changes:

```
$ svn status
> M    gizmo.rb
$ svn commit -m 'Guard against known problems'
> Sending    more_awesome/gizmo.rb
> Transmitting file data .
> Committed revision 3.
$ svn status
> ?    test
$ svn add test
```

```
> A    test
> A    test/gizmo_test.rb
$ svn commit -m 'Test coverage for problems'
> Adding    more_awesome/test
> Adding    more_awesome/test/gizmo_test.rb
> Transmitting file data .
> Committed revision 4.
```

Switching between branches

To switch between branches, you'll probably want to start with a checkout of `trunk`:

```
svn co --depth empty https://github.com/USER/REPO/trunk
```

Then, you can switch to another branch:

```
svn switch https://github.com/USER/REPO/branches/more_awesome
```

Finding the Git commit SHA for a Subversion commit

GitHub's Subversion server exposes the Git commit sha for each Subversion commit.

To see the commit SHA, you should ask for the `git-commit` unversioned remote property.

```
$ svn propget git-commit --revprop -r HEAD https://github.com/USER/REPO
05fcc584ed53d7b0c92e116cb7e64d198b13c4e3
```

With this commit SHA, you can, for example, look up the corresponding Git commit on GitHub.