Downloading source code archives

In this article

You can download a snapshot of the code in your repository.

Overview of source code archives 🔗

You can download a snapshot of any branch, tag, or specific commit from your GitHub Enterprise Server instance. These snapshots are generated by the `git archive command` in one of two formats: tarball or zipball. Snapshots don't contain the entire repository history. If you want the entire history, you can clone the repository. For more information, see "Cloning a repository."

Downloading source code archives 🔗

You can download the source code archives in three ways.

Downloading source code archives from the repository view 🔗

1. On your GitHub Enterprise Server instance, navigate to the main page of the repository.

2. Above the list of files, click <> **Code**.

Go to file	Add file ⌄	<> Code ⌄
✓ 4d5d0e1 34 minutes ago		🕒 **36,188** commits
iner.json		5 days ago
ed PR going into production		6 hours ago

3. Click 🗜 **Download ZIP**.

Downloading source code archives from a release 🔗

① On your GitHub Enterprise Server instance, navigate to the main page of the repository.

② To the right of the list of files, click **Releases**.

③ Scroll down to the "Assets" section of the release.

④ To download the source code, click ▯ **Source code (zip)** or ▯ **Source code (tar.gz)**.

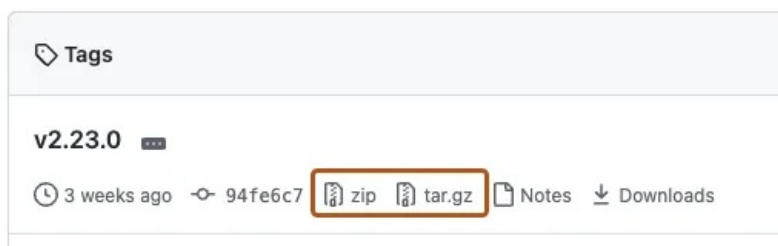## Downloading source code archives from a tag 🔗

① On your GitHub Enterprise Server instance, navigate to the main page of the repository.

② To the right of the list of files, click **Releases**.

3 At the top of the Releases page, click **Tags**.

4 To download the source code, click ⬚ **zip** or ⬚ **tar.gz**.

Source code archive URLs 🔗

Source code archives are available at specific URLs for each repository. For example, consider the repository `github/codeql`. There are different URLs for downloading a branch, a tag, or a specific commit ID.

Type of archive	Example	URL
Branch	`main`	https://github.com/github/codeql/archive/refs/**heads/main**.tar.gz
Tag	`codeql-cli/v2.12.0`	https://github.com/github/codeql/archive/refs/**tags/codeql-cli/v2.12.0**.zip
Commit	`aef66c4`	https://github.com/github/codeql/archive/**aef66c462abe817e33aad91d97aa782a1e2ad2c7**.zip

> **Note**: You can use either `.zip` or `.tar.gz` in the URLs above to request a zipball or tarball respectively.

Stability of source code archives 🔗

Source code archives are generated on request, cached for a while, and then deleted. If the same archive is requested again in the future, it'll be regenerated. It's important to understand what guarantees GitHub makes about source code archives.

- An archive of a commit ID will always have the same file contents whenever it's requested, assuming the commit ID is still in the repository and the repository's name has not changed.
- Because branches and tags can move to different commit IDs, future downloads of an archive may have different contents than previously downloaded archives of the same branch or tag. Assuming the branch or tag still points at the same commit ID, it will have the same file contents.
- The exact compression settings used to generate a zipball or tarball may change over time. The extracted contents won't change if the branch or tag doesn't change, but the outer compressed archive may have a different byte layout. GitHub will give at least six months' notice before changing compression settings.
- The name of the repository is part of the directory structure inside the archive. Therefore, if the repository name changes, the root directory name will change as

well.

If you rely on stability of source code archives for reproducibility (ensuring you always get identical files inside the archive), we recommend using the [archives REST API](#) with a commit ID for `:ref`. Using the commit ID ensures you'll always get the same file contents inside the archive and you'll be immune to repositories rewriting tags or moving branch heads.

If you rely on stability of archives for security (for example: to ensure you don't attempt to unzip a maliciously-crafted file), we recommend using releases instead of using source downloads. For more information, see "[About releases](#)."

You can use something like [this third-party GitHub action](#) to create and push these files as part of your release process. The [Release Assets REST API](#) can later be used to retrieve them.