github upload-results

In this article

Uploads a SARIF file to GitHub code scanning.

GitHub CodeQL is licensed on a per-user basis upon installation. You can use CodeQL only for certain tasks under the license restrictions. For more information, see "About the CodeQL CLI." If you have a GitHub Advanced Security license, you can use CodeQL for automated analysis, continuous integration, and continuous delivery. For more information, see "About GitHub Advanced Security."

This content describes the most recent release of the CodeQL CLI. For more information about this release, see https://github.com/github/codeql-cli-binaries/releases.

To see details of the options available for this command in an earlier release, run the command with the `--help` option in your terminal.

Synopsis

Shell

```
codeql github upload-results --sarif=<file> [--github-auth-stdin] [--github-url=<url>] [--repository=<repository-name>] [--ref=<ref>] [--commit=<commit>] [--checkout-path=<path>] <options>...
```

Description

Uploads a SARIF file to GitHub code scanning.

See: https://docs.github.com/en/code-security/secure-coding/running-codeql-cli-in-your-ci-system#uploading-results-to-github

A GitHub Apps token or personal access token must be set. For best security practices, it is recommended to set the `--github-auth-stdin` flag and pass the token to the command through standard input. Alternatively, the `GITHUB_TOKEN` environment variable can be set.

This token must have the `security_events` scope.

Options

Primary Options

`-s, --sarif=<file>`

[Mandatory] Path to the SARIF file to upload. This should be the output of codeql database analyze (or codeql database interpret-results) with `--format sarif-latest` for upload to github.com or GitHub AE, or the appropriate supported format tag for GitHub Enterprise Server instances (see https://docs.github.com/ for the right value for your release).

`-r, --repository=<repository-name>`

GitHub repository owner and name (e.g., *github/octocat*) to use as an endpoint for uploading. The CLI will attempt to autodetect this from the checkout path if it is omitted.

`-f, --ref=<ref>`

Name of the ref that was analyzed. If this ref is a pull request merge commit, then use *refs/pulls/1234/merge* or *refs/pulls/1234/head* (depending on whether or not this commit corresponds to the HEAD or MERGE commit of the PR). Otherwise, this should be a branch: *refs/heads/branch-name*. If omitted, the CLI will attempt to automatically populate this from the current branch of the checkout path, if this exists.

`-c, --commit=<commit>`

SHA of commit that was analyzed. If this is omitted the CLI will attempt to autodetect this from the checkout path.

`-p, --checkout-path=<path>`

Checkout path. Default is the current working directory.

`--merge`

[Advanced] Allow more than one SARIF file to be specified, and merge these into a single file before uploading. This is only recommended for backwards compatibility. For new analyses it is recommended to upload two separate SARIF files with different categories. This option only works in conjunction with SARIF files produced by CodeQL with SARIF version 2.1.0 (this is the default version of SARIF used by CodeQL).

`--format=<fmt>`

Select output format. Choices include:

`text` *(default)*: Print the URL for tracking the status of the SARIF upload.

`json`: Print the response body of the SARIF upload API request.

See also: https://docs.github.com/en/rest/reference/code-scanning#upload-an-analysis-as-sarif-data

Options to configure where to upload SARIF files.

`-a, --github-auth-stdin`

Accept a GitHub Apps token or personal access token via standard input.

This overrides the GITHUB_TOKEN environment variable.

`-g, --github-url=<url>`

URL of the GitHub instance to use. If omitted, the CLI will attempt to autodetect this from the checkout path and if this is not possible default to https://github.com/

Common options

`-h, --help`

Show this help text.

`-J=<opt>`

[Advanced] Give option to the JVM running the command.

(Beware that options containing spaces will not be handled correctly.)

`-v, --verbose`

Incrementally increase the number of progress messages printed.

`-q, --quiet`

Incrementally decrease the number of progress messages printed.

`--verbosity=<level>`

[Advanced] Explicitly set the verbosity level to one of errors, warnings, progress, progress+, progress++, progress+++. Overrides `-v` and `-q`.

`--logdir=<dir>`

[Advanced] Write detailed logs to one or more files in the given directory, with generated names that include timestamps and the name of the running subcommand.

(To write a log file with a name you have full control over, instead give `--log-to-stderr` and redirect stderr as desired.)