dataset import

In this article

[Plumbing] Import a set of TRAP files to a raw dataset.

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 dataset import --dbscheme=<file> [--threads=<num>] <options>... --
<dataset> <trap>...
```

Description 🔗

[Plumbing] Import a set of TRAP files to a raw dataset.

Create a dataset by populating it with TRAP files, or add data from TRAP files to an existing dataset. Updating a dataset is only possible if it has the correct dbscheme *and* its ID pool has been preserved from the initial import.

Options 🔗

Primary Options 🔗

`<dataset>` 🔗

[Mandatory] Path to the raw QL dataset to create or update. The directory will be created if it doesn't already exist.

`<trap>...` 🔗

Paths to .trap(.gz) files to import, or to directories that will be recursively scanned for .trap(.gz) files. If no files are given, an empty dataset will be created.

`-S, --dbscheme=<file>` 🔗

[Mandatory] The dbscheme definition that describes the TRAP files you want to import.

`-j, --threads=<num>` 🔗

Use this many threads for the import operation.

Defaults to 1. You can pass 0 to use one thread per core on the machine, or *-N* to leave *N* cores unused (except still use at least one thread).

`--[no-]check-undefined-labels` 🔗

[Advanced] Report errors for undefined labels.

`--[no-]check-unused-labels` 🔗

[Advanced] Report errors for unused labels.

`--[no-]check-repeated-labels` 🔗

[Advanced] Report errors for repeated labels.

`--[no-]check-redefined-labels` 🔗

[Advanced] Report errors for redefined labels.

`--[no-]check-use-before-definition` 🔗

[Advanced] Report errors for labels used before they're defined.

`--[no-]fail-on-trap-errors` 🔗

[Advanced] Exit non-zero if an error occurs during trap import.

`--[no-]include-location-in-star` 🔗

[Advanced] Construct entity IDs that encode the location in the TRAP file they came from. Can be useful for debugging of TRAP generators, but takes up a lot of space in the dataset.

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.)