

resolve tests

In this article

- Synopsis
- Description
- Options

[Deep plumbing] Find QL unit tests in given directories.

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 resolve tests <options>... -- <test|dir>...
```


Description

[Deep plumbing] Find QL unit tests in given directories.

This plumbing command is responsible for expanding the command-line parameters of subcommands that run QL unit tests, to an actual list of individual .ql and .qlref files to execute.

Options

Primary Options

`<test|dir>...` 

Each argument is one of:

- A `.ql` or `.qlref` file that defines a test to run.
- A directory which will be searched recursively for tests to run.

--slice=<N/M> [↗](#)

[Advanced] Divide the test cases into M roughly equal-sized slices and process only the $_N$ _th of them. This can be used for manual parallelization of the testing process.

--[no-]strict-test-discovery [↗](#)

[Advanced] Only use queries that can be strongly identified as tests. This mode tries to distinguish between `.ql` files that define unit tests and `.ql` files that are meant to be useful queries. This option is used by tools, such as IDEs, that need to identify all unit tests in a directory tree without depending on previous knowledge of how the files in it are arranged.

Within a QL pack whose `qlpack.yml` declares a `tests` directory, all `.ql` files in that directory are considered tests, and `.ql` files outside it are ignored. In a QL pack that doesn't declare a `tests` directory, a `.ql` file is identified as a test only if it has a corresponding `.expected` file.

For consistency, `.qlref` files are limited by the same rules as `.ql` files even though a `.qlref` file cannot really be a non-test.

--format=<fmt> [↗](#)

Select output format, either `text` (*default*) or `json`.

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.)

Legal