test run

In this article

Run unit tests for QL queries.

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 test run [--threads=<num>] [--ram=<MB>] <options>... -- <test|dir>...
```

Description

Run unit tests for QL queries.

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.

`--failing-exitcode=<code>`

[Advanced] Set the exit code to produce if any failures are encountered. Usually 1, but tooling that parses the output may find it useful to set it to 0.

`--format=<fmt>`

Select output format. Possible choices:

`text` *(default)*: A human-readable textual rendering.

`json`: A streamed JSON array of test result objects.

`betterjson`: A streamed JSON array of event objects.

`jsonz`: A stream of zero-terminated JSON test result objects.

`betterjsonz`: A stream of zero-terminated JSON event objects.

For the `betterjson` and `betterjsonz` formats, each event has a `type` property specifying the type of the event. New event types may be added in the future, so consumers should ignore any event with an unrecognized `kind` property.

`--[no-]keep-databases`

[Advanced] Preserve the databases extracted to run the test queries, even where all tests in a directory pass. (The database will always be left present when there are tests that *fail*).

`--[no-]fast-compilation`

[Deprecated] [Advanced] Omit particularly slow optimization steps when compiling test queries.

`--[no-]learn`

[Advanced] When a test produces unexpected output, instead of failing it, update its `.expected` file to match the actual output, such that it passes. Tests can still fail in this mode, for example if creation of a test database to query does not succeed.

`--consistency-queries=<dir>`

[Advanced] A directory with consistency queries that will be run for each test database. These queries should not produce any output (except when they find a problem) unless the test directory includes a `CONSISTENCY` subdirectory with a `.expected` file. This is mostly useful for testing extractors.

`--[no-]check-databases`

[Advanced] Run codeql dataset check over each test database created and report a failure if it detects inconsistencies. This is useful when testing extractors. If the check is (temporarily!) expected to fail for a particular database, place a `DB-CHECK.expected` file in the test directory.

`--[no-]show-extractor-output`

[Advanced] Show the output from extractor scripts that create test databases. This can be useful while developing or editing test cases. Beware that it can cause duplicated or malformed output if you use this with multiple threads!

`-M, --ram=<MB>`

Set total amount of RAM the test runner should be allowed to use.

`--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.

Options to find libraries and extractors used by the tests

`--search-path=<dir>[:<dir>...]`

A list of directories under which QL packs may be found. Each directory can either be a QL pack (or bundle of packs containing a `.codeqlmanifest.json` file at the root) or the immediate parent of one or more such directories.

If the path contains more than one directory, their order defines precedence between them: when a pack name that must be resolved is matched in more than one of the directory trees, the one given first wins.

Pointing this at a checkout of the open-source CodeQL repository ought to work when querying one of the languages that live there.

If you have checked out the CodeQL repository as a sibling of the unpacked CodeQL toolchain, you don't need to give this option; such sibling directories will always be searched for QL packs that cannot be found otherwise. (If this default does not work, it is strongly recommended to set up `--search-path` once and for all in a per-user configuration file).

(Note: On Windows the path separator is `;`).

`--additional-packs=<dir>[:<dir>...]`

If this list of directories is given, they will be searched for packs before the ones in `--search-path`. The order between these doesn't matter; it is an error if a pack name is found in two different places through this list.

This is useful if you're temporarily developing a new version of a pack that also appears in the default path. On the other hand, it is *not recommended* to override this option in a config file; some internal actions will add this option on the fly, overriding any configured value.

(Note: On Windows the path separator is `;`).

`--library-path=<dir>[:<dir>...]`

[Advanced] An optional list of directories that will be added to the raw import search path for QL libraries. This should only be used if you're using QL libraries that have not been packaged as QL packs.

(Note: On Windows the path separator is `;`).

`--dbscheme=<file>`

[Advanced] Explicitly define which dbscheme queries should be compiled against. This should only be given by callers that are extremely sure what they're doing.

`--compilation-cache=<dir>`

[Advanced] Specify an additional directory to use as a compilation cache.

`--no-default-compilation-cache`

[Advanced] Don't use compilation caches in standard locations such as in the QL pack containing the query or in the CodeQL toolchain directory.

Options for configuring the CodeQL package manager

`--registries-auth-stdin`

Authenticate to GitHub Enterprise Server Container registries by passing a comma-separated list of <registry_url>=<token> pairs.

For example, you can pass `https://containers.GHEHOSTNAME1/v2/=TOKEN1,https://containers.GHEHOSTNAME2/v2/=TOKEN2` to authenticate to two GitHub Enterprise Server instances.

This overrides the CODEQL_REGISTRIES_AUTH and GITHUB_TOKEN environment variables. If you only need to authenticate to the github.com Container registry, you can instead authenticate using the simpler `--github-auth-stdin` option.

`--github-auth-stdin`

Authenticate to the github.com Container registry by passing a github.com GitHub Apps token or personal access token via standard input.

To authenticate to GitHub Enterprise Server Container registries, pass `--registries-auth-stdin` or use the CODEQL_REGISTRIES_AUTH environment variable.

This overrides the GITHUB_TOKEN environment variable.

Options to control query compilation

`--no-release-compatibility`

[Advanced] Use the newest compiler features, at the cost of portability.

From time to time, new QL language features and evaluator optimizations will be supported by the QL evaluator a few releases before they are enabled by default in the QL compiler. This helps ensure that the performance you experience when developing queries in the newest CodeQL release can be matched by slightly older releases that may still be in use for Code Scanning or CI integrations.

If you do not care about your queries being compatible with other (earlier or later)

CodeQL releases, you can sometimes achieve a small amount of extra performance by using this flag to enable recent improvements in the compiler early.

In releases where there are no recent improvements to enable, this option silently does nothing. Thus it is safe to set it once and for all in your global CodeQL config file.

Available since `v2.11.1`.

Options that control the evaluation of test queries

`--[no-]tuple-counting`

[Advanced] Display tuple counts for each evaluation step in the query evaluator logs. If the `--evaluator-log` option is provided, tuple counts will be included in both the text-based and structured JSON logs produced by the command. (This can be useful for performance optimization of complex QL code).

`--timeout=<seconds>`

[Advanced] Set the timeout length for query evaluation, in seconds.

The timeout feature is intended to catch cases where a complex query would take "forever" to evaluate. It is not an effective way to limit the total amount of time the query evaluation can take. The evaluation will be allowed to continue as long as each separately timed part of the computation completes within the timeout. Currently these separately timed parts are "RA layers" of the optimized query, but that might change in the future.

If no timeout is specified, or is given as 0, no timeout will be set (except for codeql test run, where the default timeout is 5 minutes).

`-j, --threads=<num>`

Use this many threads to evaluate queries.

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).

Options for controlling outputting of structured evaluator logs

`--evaluator-log=<file>`

[Advanced] Output structured logs about evaluator performance to the given file. The format of this log file is subject to change with no notice, but will be a stream of JSON objects separated by either two newline characters (by default) or one if the `--evaluator-log-minify` option is passed. Please use `codeql generate log-summary <file>` to produce a more stable summary of this file, and avoid parsing the file directly. The file will be overwritten if it already exists.

`--evaluator-log-minify`

[Advanced] If the `--evaluator-log` option is passed, also passing this option will minimize the size of the JSON log produced, at the expense of making it much less human readable.

Options for checking imported TRAP

`--[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.)