query compile

In this article

Compile or check QL code.

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 query compile [--check-only] [--keep-going] [--threads=<num>] [--ram=<MB>]
<options>... -- <file>...
```

Description

Compile or check QL code.

Compile one or more queries. Usually the main outcome of this command is that the compiled version of the query is written to a *compilation cache* where it will be found when the query is later executed. Other output options are mostly for debugging.

Options

Primary Options

`<file>...`

[Mandatory] Queries to compile. Each argument is one of:

- A .ql file to compile.

- A directory which will be searched recursively for .ql files.
- A .qls file that defines a particular set of queries.
- The basename of a "well-known" .qls file exported by one of the installed QL packs.

`-n, --check-only`

Just check that the QL is valid and print any errors; do not actually optimize and store a query plan. This can be much faster than a full compilation.

`--[no-]precompile`

[Advanced] Save each compiled query as a binary `.qlx` file next to the `.ql` source.

This is only supposed to be used while preparing a query pack for distribution (in which case it is used automatically by codeql pack publish). Once the `.qlx` files exist, later commands that execute queries may ignore changes to the QL source in favor of the precompiled version.

Some rarely used compilation options are incompatible with this and will lead to a run-time error.

Available since `v2.12.0`.

`--[no-]dump-dil`

[Advanced] Print the optimized DIL intermediate representation to standard output while compiling.

When JSON output is selected, the DIL will be represented as an array of single-line strings, with some wrapping to identify which query is being compiled.

`-k, --[no-]keep-going`

Keep going with compilation even if an error is found.

`--[no-]dump-ra`

[Advanced] Print the optimized RA query plan to standard output while compiling.

When JSON output is selected, the RA will be represented as an array of single-line strings, with some wrapping to identify which query is being compiled.

`--format=<fmt>`

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

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

Use this many threads to compile 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).

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

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

QL variant and compiler control options

`--warnings=<mode>`

How to handle warnings from the QL compiler. One of:

`hide`: Suppress warnings.

`show` *(default)*: Print warnings but continue with compilation.

`error`: Treat warnings as errors.

`--no-debug-info`

Don't emit source location info in RA for debugging.

`--[no-]fast-compilation`

[Deprecated] [Advanced] Omit particularly slow optimization steps.

`--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`.

`--[no-]local-checking`

Only perform initial checks on the part of the QL source that is used.

`--no-metadata-verification`

Don't check embedded query metadata in QLDoc comments for validity.

`--compilation-cache-size=<MB>`

[Advanced] Override the default maximum size for a compilation cache directory.

Options to set up compilation environment

`--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.

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.)