database analyze

In this article

Analyze a database, producing meaningful results in the context of the source 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 database analyze --format=<format> --output=<output> [--threads=<num>] [--
ram=<MB>] <options>... -- <database> <query|dir|suite|pack>...
```

Description

Analyze a database, producing meaningful results in the context of the source code.

Run a query suite (or some individual queries) against a CodeQL database, producing results, styled as alerts or paths, in SARIF or another interpreted format.

This command combines the effect of the codeql database run-queries and codeql database interpret-results commands. If you want to run queries whose results *don't* meet the requirements for being interpreted as source-code alerts, use codeql database run-queries or codeql query run instead, and then codeql bqrs decode to convert the raw results to a readable notation.

Options

Primary Options

`<database>`

[Mandatory] Path to the CodeQL database to query.

`<querysuite|pack>...`

Queries to execute. Each argument is in the form `scope/name@range:path` where:

- `scope/name` is the qualified name of a CodeQL pack.
- `range` is a semver range.
- `path` is a file system path.

If a `scope/name` is specified, the `range` and `path` are optional. A missing `range` implies the latest version of the specified pack. A missing `path` implies the default query suite of the specified pack.

The `path` can be one of a `*.ql` query file, a directory containing one or more queries, or a `.qls` query suite file. If there is no pack name specified, then a `path` must be provided, and will be interpreted relative to the current working directory of the current process.

To specify a `path` that contains a literal `@` or `:`, use `path:` as a prefix to the argument, like this: `path:directory/with:and@/chars`.

If a `scope/name` and `path` are specified, then the `path` cannot be absolute. It is considered relative to the root of the CodeQL pack.

If no queries are specified, the CLI will automatically determine a suitable set of queries to run. In particular, if a Code Scanning configuration file was specified at database creation time using `--codescanning-config` then the queries from this will be used. Otherwise, the default queries for the language being analyzed will be used.

`--format=<format>`

[Mandatory] The format in which to write the results. One of:

`csv`: Formatted comma-separated values, including columns with both rule and alert metadata.

`sarif-latest`: Static Analysis Results Interchange Format (SARIF), a JSON-based format for describing static analysis results. This format option uses the most recent supported version (v2.1.0). This option is not suitable for use in automation as it will produce different versions of SARIF between different CodeQL versions.

`sarifv2.1.0`: SARIF v2.1.0.

`graphtext`: A textual format representing a graph. Only compatible with queries with @kind graph.

`dgml`: Directed Graph Markup Language, an XML-based format for describing graphs. Only compatible with queries with @kind graph.

`dot`: Graphviz DOT language, a text-based format for describing graphs. Only compatible with queries with @kind graph.

`-o, --output=<output>`

[Mandatory] The output path to write results to. For graph formats this should be a directory, and the result (or results if this command supports interpreting more than one query) will be written within that directory.

`--[no-]rerun`

Evaluate even queries that seem to have a BQRS result stored in the database already.

`--no-print-diagnostics-summary`

Don't print a summary of the analyzed diagnostics to standard output.

`--no-print-metrics-summary`

Don't print a summary of the analyzed metrics to standard output.

`--max-paths=<maxPaths>`

The maximum number of paths to produce for each alert with paths. (Default: 4)

`--[no-]sarif-add-file-contents`

[SARIF formats only] Include the full file contents for all files referenced in at least one result.

`--[no-]sarif-add-snippets`

[SARIF formats only] Include code snippets for each location mentioned in the results, with two lines of context before and after the reported location.

`--[no-]sarif-add-query-help`

[SARIF formats only] Include Markdown query help in the results. It loads query help for /path/to/query.ql from the /path/to/query.md file. This option has no effect when passed to codeql bqrs interpret.

`--[no-]sarif-group-rules-by-pack`

[SARIF formats only] Place the rule object for each query under its corresponding QL pack in the `<run>.tool.extensions` property. This option has no effect when passed to codeql bqrs interpret.

`--[no-]sarif-multicause-markdown`

[SARIF formats only] For alerts that have multiple causes, include them as a Markdown-formatted itemized list in the output in addition to as a plain string.

`--no-group-results`

[SARIF formats only] Produce one result per message, rather than one result per unique location.

`--csv-location-format=<csvLocationFormat>`

The format in which to produce locations in CSV output. One of: uri, line-column, offset-length. (Default: line-column)

`--dot-location-url-format=<dotLocationUrlFormat>`

A format string defining the format in which to produce file location URLs in DOT output.

The following place holders can be used {path} {start:line} {start:column} {end:line} {end:column}, {offset}, {length}

`--sarif-category=<category>`

[SARIF formats only] Specify a category for this analysis to include in the SARIF output. A category can be used to distinguish multiple analyses performed on the same commit and repository, but on different languages or different parts of the code.

If you analyze the same version of a code base in several different ways (e.g., for different languages) and upload the results to GitHub for presentation in Code Scanning, this value should differ between each of the analyses, which tells Code Scanning that the analyses *supplement* rather than *supersede* each other. (The values should be consistent between runs of the same analysis for *different* versions of the code base.)

This value will appear (with a trailing slash appended if not already present) as the `<run>.automationId` property in SARIF v1, the `<run>.automationLogicalId` property in SARIF v2, and the `<run>.automationDetails.id` property in SARIF v2.1.0.

`--[no-]download`

Download any missing queries before analyzing.

Options to control the query evaluator

`--[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).

`--[no-]save-cache`

[Advanced] Aggressively write intermediate results to the disk cache. This takes more time and uses (much) more disk space, but may speed up the subsequent execution of similar queries.

`--[no-]expect-discarded-cache`

[Advanced] Make decisions about which predicates to evaluate, and what to write to the disk cache, based on the assumption that the cache will be discarded after the queries have been executed.

`--[no-]keep-full-cache`

[Advanced] Don't clean up the disk cache after evaluation completes. This may save time if you're going to do codeql dataset cleanup or codeql database cleanup afterwards anyway.

`--max-disk-cache=<MB>`

Set the maximum amount of space that the disk cache for intermediate query results can use.

If this size is not configured explicitly, the evaluator will try to use a "reasonable" amount of cache space, based on the size of the dataset and the complexity of the queries. Explicitly setting a higher limit than this default usage will enable additional caching which can speed up later queries.

`--min-disk-free=<MB>`

[Advanced] Set target amount of free space on file system.

If `--max-disk-cache` is not given, the evaluator will try hard to curtail disk cache usage if the free space on the file system drops below this value.

`--min-disk-free-pct=<pct>`

[Advanced] Set target fraction of free space on file system.

If `--max-disk-cache` is not given, the evaluator will try hard to curtail disk cache usage if the free space on the file system drops below this percentage.

`--external=<pred>=<file.csv>`

A CSV file that contains rows for external predicate *<pred>*. Multiple `--external` options can be supplied.

`--xterm-progress=<mode>`

[Advanced] Controls whether to show progress tracking during QL evaluation using xterm control sequences. Possible values are:

`no`: Never produce fancy progress; assume a dumb terminal.

`auto` *(default)*: Autodetect whether the command is running in an appropriate terminal.

`yes`: Assume the terminal can understand xterm control sequences. The feature still depends on being able to autodetect the *size* of the terminal, and will also be disabled if `-q` is given.

`25x80` (or similar): Like `yes`, and also explicitly give the size of the terminal.

`25x80:/dev/pts/17` (or similar): show fancy progress on a *different* terminal than stderr. Mostly useful for internal testing.

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 to control RAM usage

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

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

Options to control QL compilation

`--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.)