Using custom queries with the CodeQL CLI

In this article

You can write your own CodeQL queries to find specific vulnerabilities and errors.

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."

About custom queries and the CodeQL CLI

You can customize your CodeQL analyses by writing your own queries to highlight specific vulnerabilities or errors.

This topic is specifically about writing queries to use with the database analyze command to produce interpreted results.

Note: Queries run with `database analyze` have strict metadata requirements. You can also execute queries using the following plumbing-level subcommands:

- database run-queries, which outputs non-interpreted results in an intermediate binary format called BQRS.
- query run, which will output BQRS files, or print results tables directly to the command line. Viewing results directly in the command line may be useful for iterative query development using the CLI.

Queries run with these commands don't have the same metadata requirements. However, to save human-readable data you have to process each BQRS results file using the bqrs decode plumbing subcommand. Therefore, for most use cases it's easiest to use database analyze to directly generate interpreted results.

Writing a valid query

Before running a custom analysis you need to write a valid query, and save it in a file with a `.ql` extension. There is extensive documentation available to help you write

queries. For more information, see "CodeQL queries."

Including query metadata

Query metadata is included at the top of each query file. It provides users with information about the query, and tells the CodeQL CLI how to process the query results.

When running queries with the `database analyze` command, you must include the following two properties to ensure that the results are interpreted correctly:

- Query identifier (`@id`): a sequence of words composed of lowercase letters or digits, delimited by `/` or `-`, identifying and classifying the query.
- Query type (`@kind`): identifies the query as a simple alert (`@kind problem`), an alert documented by a sequence of code locations (`@kind path-problem`), for extractor troubleshooting (`@kind diagnostic`), or a summary metric (`@kind metric` and `@tags summary`).

For more information about these metadata properties, see "Metadata for CodeQL queries" and the Query metadata style guide.

> **Note:** Metadata requirements may differ if you want to use your query with other applications. For more information, see "Metadata for CodeQL queries."

Packaging custom QL queries

> **Note:** The CodeQL package management functionality, including CodeQL packs, is currently available as a beta release and is subject to change. During the beta release, CodeQL packs are available only using GitHub Packages - the Container registry. To use this beta functionality, install the latest version of the CodeQL CLI bundle from: https://github.com/github/codeql-action/releases.

When you write your own queries with the intention to share them with others, you should save them in a custom CodeQL pack. You can publish the pack as a CodeQL pack to GitHub Packages - the GitHub Container registry. For more information, see "Customizing analysis with CodeQL packs."

CodeQL packs organize the files used in CodeQL analysis and can store queries, library files, query suites, and important metadata. Their root directory must contain a file named `qlpack.yml`. Your custom queries should be saved in the CodeQL pack root, or its subdirectories.

For each CodeQL pack, the `qlpack.yml` file includes information that tells the CodeQL CLI how to compile the queries, which other CodeQL packs and libraries the pack depends on, and where to find query suite definitions. For more information about what to include in this file, see "Customizing analysis with CodeQL packs."

Including query help for custom CodeQL queries in SARIF files

If you use the CodeQL CLI to run code scanning analyses on third party CI/CD systems, you can include the query help for your custom queries in SARIF files generated during an analysis. After uploading the SARIF file to GitHub, the query help is shown in the code scanning UI for any alerts generated by the custom queries.

From CodeQL CLI v2.7.1 onwards, you can include markdown-rendered query help in SARIF files by providing the `--sarif-add-query-help` option when running `codeql`

`database analyze`.

You can write query help for custom queries directly in a markdown file and save it alongside the corresponding query. Alternatively, for consistency with the standard CodeQL queries, you can write query help in the `.qhelp` format. Query help written in `.qhelp` files can't be included in SARIF files, and they can't be processed by code scanning so must be converted to markdown before running the analysis. For more information, see "Query help files" and "Testing query help files."

Contributing to the CodeQL repository

If you would like to share your query with other CodeQL users, you can open a pull request in the CodeQL repository. For more information, see Contributing to CodeQL.

Further reading

- "CodeQL queries"