Customizing analysis with CodeQL packs

In this article

You can use CodeQL packs to run CodeQL queries maintained by other people, or to share CodeQL queries that you've developed.

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."

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.

About CodeQL packs

CodeQL packs are used to create, share, depend on, and run CodeQL queries and libraries. CodeQL packs contain queries, library files, query suites, and metadata. You can customize your CodeQL analysis by downloading packs created by others and running them on your codebase.

There are three types of CodeQL packs: query packs, library packs, and model packs.

- Query packs contain a set of pre-compiled queries that can be evaluated on a CodeQL database. Query packs are designed to be run. When a query pack is published, the bundle includes all the transitive dependencies and pre-compiled representations of each query, in addition to the query sources. This ensures consistent and efficient execution of the queries in the pack.
- Library packs are designed to be used by query packs (or other library packs) and do not contain queries themselves. The libraries are not compiled separately.
- Model packs can be used to expand code scanning analysis to recognize libraries and frameworks that are not supported by default. Model packs are currently in beta and subject to change. During the beta, model packs are available for Java analysis at the repository level. For more information about creating your own model packs, see "Creating and working with CodeQL packs."

The standard CodeQL packs for all supported languages are published in the Container registry. If you installed the CodeQL CLI in the standard way, using the CodeQL CLI bundle, the core query packs are already downloaded and available to you. They are:

- `codeql/cpp-queries`
- `codeql/csharp-queries`
- `codeql/go-queries`
- `codeql/java-queries`
- `codeql/javascript-queries`
- `codeql/python-queries`
- `codeql/ruby-queries`

You can also use the CodeQL CLI to create your own CodeQL packs, add dependencies to packs, and install or update dependencies. For more information, see "Creating and working with CodeQL packs."

You can publish CodeQL packs that you have created, using the CodeQL CLI. For more information on publishing and downloading CodeQL packs, see "Publishing and using CodeQL packs."

Downloading and using CodeQL query packs

The CodeQL CLI bundle includes queries that are maintained by GitHub experts, security researchers, and community contributors. If you want to run queries developed by other organizations, CodeQL query packs provide an efficient and reliable way to download and run queries, while model packs (beta) can be used to expand code scanning analysis to recognize libraries and frameworks that are not supported by default. For more information about query packs, see "About code scanning with CodeQL." For information about writing your own model packs, see "Creating and working with CodeQL packs."

Before you can use a CodeQL query pack to analyze a database, you must download any packages you require from the GitHub Container registry. This can be done either by using the `--download` flag as part of the `codeql database analyze` command, or running `codeql pack download`. If a package is not publicly available, you will need to use a GitHub App or personal access token to authenticate. For more information and an example, see "Uploading CodeQL analysis results to GitHub."

Option	Required	Usage
`<scope/name@version:path>`	✓	Specify the scope and name of one or more CodeQL query packs to download using a comma-separated list. Optionally, include the version to download and unzip. By default the latest version of this pack is downloaded. Optionally, include a path to a query, directory, or query suite to run. If no path is included, then run the default queries of this pack.
`--github-auth-stdin`	×	Pass the CLI the GitHub App or personal access token created for authentication with GitHub's REST API from your secret store via standard input. This is not needed if the command has access to a `GITHUB_TOKEN` environment variable set with this token.

Note: If you specify a particular version of a query pack to use, be aware that the version you specify may eventually become too old for the latest version of CodeQL to make efficient use of. To ensure optimal performance, if you need to specify exact query pack versions, you should reevaluate which versions you pin to whenever you upgrade the CodeQL CLI you're using.

For more information about pack compatibility, see "Publishing and using CodeQL packs."

Basic example of downloading and using query packs

This example runs the `codeql database analyze` command with the `--download` option to:

1. Download the latest version of the `octo-org/security-queries` pack.
2. Download a version of the `octo-org/optional-security-queries` pack that is *compatible* with version 1.0.1 (in this case, it is version 1.0.2). For more information on semver compatibility, see npm's semantic version range documentation.
3. Run all the default queries in `octo-org/security-queries`.
4. Run only the query `queries/csrf.ql` from `octo-org/optional-security-queries`

```
$ echo $OCTO-ORG_ACCESS_TOKEN | codeql database analyze --download /codeql-
dbs/example-repo \
    octo-org/security-queries \
    octo-org/optional-security-queries@~1.0.1:queries/csrf.ql \
    --format=sarif-latest --output=/temp/example-repo-js.sarif

> Download location: /Users/mona/.codeql/packages
> Installed fresh octo-org/security-queries@1.0.0
> Installed fresh octo-org/optional-security-queries@1.0.2
> Running queries.
> Compiling query plan for /Users/mona/.codeql/packages/octo-org/security-
queries/1.0.0/potential-sql-injection.ql.
> [1/2] Found in cache: /Users/mona/.codeql/packages/octo-org/security-
queries/1.0.0/potential-sql-injection.ql.
> Starting evaluation of octo-org/security-queries/query1.ql.
> Compiling query plan for /Users/mona/.codeql/packages/octo-org/optional-
security-queries/1.0.2/queries/csrf.ql.
> [2/2] Found in cache: /Users/mona/.codeql/packages/octo-org/optional-security-
queries/1.0.2/queries/csrf.ql.
> Starting evaluation of octo-org/optional-security-queries/queries/csrf.ql.
> [2/2 eval 694ms] Evaluation done; writing results to octo-org/security-
queries/query1.bqrs.
> Shutting down query evaluator.
> Interpreting results.
```

Direct download of CodeQL packs

If you want to download a CodeQL pack without running it immediately, then you can use the `codeql pack download` command. This is useful if you want to avoid accessing the internet when running CodeQL queries. When you run the CodeQL analysis, you can specify packs, versions, and paths in the same way as in the previous example:

```
echo $OCTO-ORG_ACCESS_TOKEN | codeql pack download <scope/name@version:path>
<scope/name@version:path> ...
```

Downloading CodeQL packs from multiple GitHub container

registries

If your CodeQL packs reside on multiple container registries, then you must instruct the CodeQL CLI where to find each pack. For more information, see "Customizing your advanced setup for code scanning."

Specifying which queries to run in a CodeQL pack

Query specifiers are used by `codeql database analyze` and other commands that operate on a set of queries. The complete form of a query specifier is `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 to a single query, a directory containing queries, or a query suite file.

When you specify a `scope/name`, the `range` and `path` are optional. If you omit a `range` then the latest version of the specified pack is used. If you omit a `path` then the default query suite of the specified pack is used.

The `path` can be one of: a `.ql` query file, a directory containing one or more queries, or a `.qls` query suite file. If you omit a pack name, then you must provide a `path`, which will be interpreted relative to the working directory of the current process. Glob patterns are not supported.

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

Example query specifiers

- `codeql/python-queries` - All the queries in the default query suite of the latest version of the `codeql/python-queries` pack.
- `codeql/python-queries@1.2.3` - All the queries in the default query suite of version `1.2.3` of the `codeql/python-queries` pack.
- `codeql/python-queries@~1.2.3` - All the queries in the default query suite of the latest version of the `codeql/python-queries` pack that is >= `1.2.3` and < `1.3.0`.
- `codeql/python-queries:Functions` - All queries in the `Functions` directory in the latest version of the `codeql/python-queries` pack.
- `codeql/python-queries@1.2.3:Functions` - All queries in the `Functions` directory in version 1.2.3 of the `codeql/python-queries` pack.
- `codeql/python-queries@1.2.3:codeql-suites/python-code-scanning.qls` - All queries in the `codeql-suites/python-code-scanning.qls` directory in version 1.2.3 of the `codeql/python-queries` pack.
- `suites/my-suite.qls` - All queries in the `suites/my-suite.qls` file relative to the current working directory.

Tip

The default query suite of the standard CodeQL query packs are `codeql-suites/<lang>-code-scanning.qls`. Several other useful query suites can also be found in the `codeql-suites` directory of each pack. For example, the `codeql/cpp-queries` pack contains the following query suites:

- `cpp-code-scanning.qls` - Standard Code Scanning queries for C++. The default query suite for this pack.

- `cpp-security-extended.qls` - Queries from the default `cpp-code-scanning.qls` suite for C++, plus lower severity and precision queries.
- `cpp-security-and-quality.qls` - Queries from `cpp-security-extended.qls`, plus maintainability and reliability queries.

You can see the sources for these query suites in the CodeQL repository. Query suites for other languages are similar.

Using model packs to analyze calls to custom dependencies

You can include published model packs in a code scanning analysis with the `--model-packs` option. For example:

```
$ codeql database analyze /codeql-dbs/my-company --format=sarif-latest \
  --model-packs my-repo/my-java-model-pack \
  --output=/temp/my-company.sarif codeql/java-queries
```

In this example, the relevant queries in the standard query pack `codeql/java-queries` will use the dependency information from the model pack, `my-repo/my-java-model-pack`, to check for vulnerabilities in code that calls those dependencies.

You can specify multiple published model packs in an analysis.

For more information about writing your own model packs, see "Creating and working with CodeQL packs."

About published packs

When a pack is published for use in analyses, the `codeql pack create` or `codeql pack publish` command verifies that the content is complete and also adds some additional pieces of content to it:

- For query packs, a copy of each of the library packs it depends on, in the precise versions it has been developed with. Users of the query pack won't need to download these library packs separately.
- For query packs, precompiled representations of each of the queries. These are faster to execute than it would be to compile the QL source for the query at each analysis.

Most of this data is located in a directory named `.codeql` in the published pack, but precompiled queries are in files with a `.qlx` suffix next to the `.ql` source for each query. When analyzing a database with a query from a published pack, CodeQL will load these files instead of the `.ql` source. If you need to modify the content of a *published* pack, be sure to remove all of the `.qlx` files, since they may prevent modifications in the `.ql` files from taking effect.