

database init

In this article

- Synopsis
- Description
- Options

[Plumbing] Create an empty CodeQL database.


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 init --source-root=<dir> [--language=<lang>[,<lang>...]] [--github-auth-stdin] [--github-url=<url>] [--extractor-option=<extractor-option-name=value>] <options>... -- <database>
```

Description

[Plumbing] Create an empty CodeQL database.

Create a skeleton structure for a CodeQL database that doesn't have a raw QL dataset yet, but is ready for running extractor steps. After this command completes, run one or more [codeql database trace-command](#) commands followed by [codeql database finalize](#) to prepare the database for querying.

(Part of what this does is resolve the location of the appropriate language pack and store it in the database metadata, such that it won't need to be redone at each extraction command. It is not valid to switch extractors in the middle of an extraction operation anyway.)

Options

Primary Options

`<database>`

[Mandatory] Path to the CodeQL database to create. This directory will be created, and *must not* already exist (but its parent must).

If the `--db-cluster` option is given, this will not be a database itself, but a directory that will *contain* databases for several languages built from the same source root.

It is important that this directory is not in a location that the build process will interfere with. For instance, the `target` directory of a Maven project would not be a suitable choice.

`-s, --source-root=<dir>`

[Mandatory] The root source code directory. In many cases, this will be the checkout root. Files within it are considered to be the primary source files for this database. In some output formats, files will be referred to by their relative path from this directory.

`--[no-]overwrite`

[Advanced] If the database already exists, delete it and proceed with this command instead of failing. This option should be used with caution as it may recursively delete the entire database directory.

`--codescanning-config=<file>`

[Advanced] Read a Code Scanning configuration file specifying options on how to create the CodeQL databases and what queries to run in later steps. For more details on the format of this configuration file, refer to [Customizing your advanced setup for code scanning](#). To run queries from this file in a later step, invoke [codeql database analyze](#) without any other queries specified.

`--[no-]db-cluster`

Instead of creating a single database, create a "cluster" of databases for different languages, each of which is a subdirectory of the directory given on the command line.

`-l, --language=<lang>[,<lang>...]`

The language that the new database will be used to analyze.


Use [codeql resolve languages](#) to get a list of the pluggable language extractors found on the search path.

When the `--db-cluster` option is given, this can appear multiple times, or the value can be a comma-separated list of languages.

If this option is omitted, and the source root being analysed is a checkout of a GitHub repository, the CodeQL CLI will make a call to the GitHub API to attempt to automatically determine what languages to analyse. Note that to be able to do this, a GitHub PAT token must be supplied either in the environment variable `GITHUB_TOKEN` or via standard input using the `--github-auth-stdin` option.

`--[no-]allow-missing-source-root`

[Advanced] Proceed even if the specified source root does not exist.

`--[no-]begin-tracing` 


[Advanced] Create some scripts that can be used to set up "indirect build tracing," which allows integration into existing build workflows when an explicit build command is not available. For information about when and how to use this feature, please refer to our documentation at [Preparing your code for CodeQL analysis](#).

Baseline calculation options

`--[no-]calculate-baseline` 

[Advanced] Calculate baseline information about the code being analyzed and add it to the database. By default, this is enabled unless the source root is the root of a filesystem. This flag can be used to either disable, or force the behavior to be enabled even in the root of the filesystem.

Extractor selection options

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

A list of directories under which extractor packs may be found. The directories can either be the extractor packs themselves or directories that contain extractors as immediate subdirectories.

If the path contains multiple directory trees, their order defines precedence between them: if the target language is matched in more than one of the directory trees, the one given first wins.

The extractors bundled with the CodeQL toolchain itself will always be found, but if you need to use separately distributed extractors you need to give this option (or, better yet, set up `--search-path` in a per-user configuration file).


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

Options to configure how to call the GitHub API to auto-detect languages.

`-a, --github-auth-stdin` 

Accept a GitHub Apps token or personal access token via standard input.

This overrides the `GITHUB_TOKEN` environment variable.

`-g, --github-url=<url>` 

URL of the GitHub instance to use. If omitted, the CLI will attempt to autodetect this from the checkout path and if this is not possible default to <https://github.com/>

Options to configure the 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.

Options to configure Windows tracing [↗](#)

`--trace-process-name=<process-name>` [↗](#)

[Windows only] When initializing tracing, inject the tracer into a parent process of the CodeQL CLI whose name matches this argument. If more than one parent process has this name, the one lowest in the process tree will be selected. This option overrides `--trace-process-level`, so if both are passed, only this option will be used.

`--trace-process-level=<process-level>` [↗](#)

[Windows only] When initializing tracing, inject the tracer this many parents above the current process, with 0 corresponding to the process that is invoking the CodeQL CLI. The CLI's default behavior if no arguments are passed is to inject into the parent of the calling process, with some special cases for GitHub Actions and Azure Pipelines.

Options to configure indirect build tracing [↗](#)

`--no-tracing` [↗](#)

[Advanced] Do not trace the specified command, instead rely on it to produce all necessary data directly.

`--extra-tracing-config=<tracing-config.lua>` [↗](#)

[Advanced] The path to a tracer configuration file. It may be used to modify the behaviour of the build tracer. It may be used to pick out compiler processes that run as part of the build command, and trigger the execution of other tools. The extractors will provide default tracer configuration files that should work in most situations.

Options to control extractor behavior: only be applied to the indirect tracing environment [↗](#)

`-0, --extractor-option=<extractor-option-name=value>` [↗](#)


Set options for CodeQL extractors. `extractor-option-name` should be of the form `extractor_name.group1.group2.option_name` or `group1.group2.option_name`. If `extractor_option_name` starts with an extractor name, the indicated extractor must declare the option `group1.group2.option_name`. Otherwise, any extractor that declares the option `group1.group2.option_name` will have the option set. `value` can be any string that does not contain a newline.

You can use this command-line option repeatedly to set multiple extractor options. If you provide multiple values for the same extractor option, the behaviour depends on the type that the extractor option expects. String options will use the last value provided. Array options will use all the values provided, in order. Extractor options specified using this command-line option are processed after extractor options given via `--extractor-options-file`.

When passed to [codeql database init](#) or `codeql database begin-tracing`, the options will

only be applied to the indirect tracing environment. If your workflow also makes calls to [codeql database trace-command](#) then the options also need to be passed there if desired.

See <https://codeql.github.com/docs/codeql-cli/extractor-options> for more information on CodeQL extractor options, including how to list the options declared by each extractor.

--extractor-options-file=<extractor-options-bundle-file> 


Specify extractor option bundle files. An extractor option bundle file is a JSON file (extension `.json`) or YAML file (extension `.yaml` or `.yml`) that sets extractor options. The file must have the top-level map key 'extractor' and, under it, extractor names as second-level map keys. Further levels of maps represent nested extractor groups, and string and array options are map entries with string and array values.

Extractor option bundle files are read in the order they are specified. If different extractor option bundle files specify the same extractor option, the behaviour depends on the type that the extractor option expects. String options will use the last value provided. Array options will use all the values provided, in order. Extractor options specified using this command-line option are processed before extractor options given via `--extractor-option`.

When passed to [codeql database init](#) or `codeql database begin-tracing`, the options will only be applied to the indirect tracing environment. If your workflow also makes calls to [codeql database trace-command](#) then the options also need to be passed there if desired.

See <https://codeql.github.com/docs/codeql-cli/extractor-options> for more information on CodeQL extractor options, including how to list the options declared by each extractor.

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.)

Legal

© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Status](#) [Pricing](#) [Expert services](#) [Blog](#)