Creating and working with CodeQL packs

In this article

You can use CodeQL packs to create, share, depend on, and run CodeQL queries and libraries.

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 and the CodeQL CLI

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. With CodeQL packs and the package management commands in the CodeQL CLI, you can publish your custom queries and integrate them into your codebase analysis.

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

- 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 include dependencies 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

"CodeQL."

You can use the `pack` command in the CodeQL CLI to create CodeQL packs, add dependencies to packs, and install or update dependencies. You can also publish and download CodeQL packs using the `pack` command. For more information, see "Publishing and using CodeQL packs."

For more information about compatibility between published query packs and different CodeQL releases, see "Publishing and using CodeQL packs."

The standard CodeQL packages for all supported languages are published in the Container registry. The CodeQL repository contains source files for the standard CodeQL packs for all supported languages. The core query packs, which are included in the CodeQL CLI bundle, but you can otherwise download, are:

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

CodeQL pack structure

A CodeQL pack must contain a file called `qlpack.yml` in its root directory. In the `qlpack.yml` file, the `name:` field must have a value that follows the format of `<scope>/<pack>`, where `<scope>` is the GitHub organization or user account that the pack will be published to and `<pack>` is the name of the pack. Additionally, query packs and library packs with CodeQL tests contain a `codeql-pack.lock.yml` file that contains the resolved dependencies of the pack. This file is generated during a call to the `codeql pack install` command, is not meant to be edited by hand, and should be added to your version control system.

The other files and directories within the pack should be logically organized. For example, typically:

- Queries are organized into directories for specific categories.
- Queries for specific products, libraries, and frameworks are organized into their own top-level directories.

Creating a CodeQL pack

You can create a CodeQL pack by running the following command from the checkout root of your project:

```
codeql pack init <scope>/<pack>
```

You must specify:

- `<scope>`: the name of the GitHub organization or user account that you will publish to.
- `<pack>`: the name for the pack that you are creating.

The `codeql pack init` command creates the directory structure and configuration files for a CodeQL pack. By default, the command creates a query pack. If you want to create a library pack, you must edit the `qlpack.yml` file to explicitly declare the file as a library

pack by including the `library:true` property.

Creating a CodeQL model pack

> **Note:** Model packs are currently in beta and subject to change. During the beta, model packs are supported only by Java analysis.

Model packs can be used to expand code scanning analysis to recognize libraries and frameworks that are not supported by default. Model packs use data extensions, which are implemented as YAML and describe how to add data for new dependencies. When a model pack is specified, the data extensions in that pack will be added to the code scanning analysis automatically. For more information about CodeQL model packs and data extensions, see Using the CodeQL model editor in the CodeQL documentation.

A model pack is a CodeQL pack with the following characteristics in the `qlpack.yml` file:

- It defines `library: true`.
- It has no dependencies.
- It has one or more `extensionTargets`.
- It has a `dataExtensions` property that points to one or more data extension files.

A model pack will inject its specified data extensions into each query pack that is named in `extensionTargets`, if it falls within the specified version range. For example:

```
name: my-repo/my-java-model-pack
version: 1.2.3
extensionTargets:
  codeql/java-all: ~1.2.3
  codeql/util: ~4.5.6
dataExtensions:
  - models/**/*.yml
```

In this example, the model pack will inject all the data extensions in `models/**/` into a `codeql/java-all` query pack that is at a version from `1.2.3` up to and including `1.3.0`, and a `codeql/util` query pack that is at a version from `4.5.6` up to and including `4.6.0`. For more information, see "Using semantic versioning" in the npm documentation and the "Semantic versioning specification."

Once you've created a model pack, you can publish it in the same way as other CodeQL packs. For more information, see "Publishing and using CodeQL packs." You can then use published model packs in a code scanning analysis with the `--model-packs` option. For more information, see "Customizing analysis with CodeQL packs."

Modifying an existing legacy QL pack to create a CodeQL query pack

If you already have a `qlpack.yml` file, you can edit it manually to convert it into a CodeQL pack.

1. Edit the `name` property so that it matches the format `<scope>/<name>`, where `<scope>` is the name of the GitHub organization or user account that you will publish to.
2. In the `qlpack.yml` file, include a `version` property with a semver identifier, as well as an optional `dependencies` block.
3. Migrate the list of dependencies in `libraryPathDependencies` to the `dependencies` block. Specify the version range for each dependency. If the range is unimportant,

or you are unsure of compatibility, you can specify `"\*"`, which indicates that any version is acceptable and will default to the latest version when you run `codeql pack install`.

For more information about the properties, see "Customizing analysis with CodeQL packs."

Adding and installing dependencies on a CodeQL pack

Note: This is only supported for CodeQL query and library packs.

You can add dependencies on CodeQL packs using the command `codeql pack add`. You must specify the scope, name, and (optionally) a compatible version range.

```
codeql pack add <scope>/<name>@x.x.x <scope>/<other-name>
```

If you don’t specify a version range, the latest version will be added. Otherwise, the latest version that satisfies the requested range will be added.

This command updates the `qlpack.yml` file with the requested dependencies and downloads them into the package cache. Please note that this command will reformat the file and remove all comments.

You can also manually edit the `qlpack.yml` file to include dependencies and install the dependencies with the command:

```
codeql pack install
```

This command downloads all dependencies to the shared cache on the local disk.

Notes:

- Running the `codeql pack add` and `codeql pack install` commands will generate or update the `codeql-pack.lock.yml` file. This file should be checked-in to version control. The `codeql-pack.lock.yml` file contains the precise version numbers used by the pack. For more information, see "About codeql-pack.lock.yml files."
- By default `codeql pack install` will install dependencies from the Container registry on GitHub.com. You can install dependencies from a GitHub Enterprise Server Container registry by creating a `qlconfig.yml` file. For more information, see "Publishing and using CodeQL packs" in the GitHub Enterprise Server documentation.

Customizing a downloaded CodeQL pack

The recommended way to experiment with changes to a pack is to clone the repository containing its source code.

If no source repository is available and you need to base modifications on a pack downloaded from the Container registry, be aware that these packs are not intended to be modified or customized after downloading, and their format may change in the future without much notice. We recommend taking the following steps after downloading a pack if you need to modify the content:

- Change the pack *name* in `qlpack.yml` so you avoid confusion with results from the unmodified pack.

- Remove all files named `*.qlx` anywhere in the unpacked directory structure. These files contain precompiled versions of the queries, and in some situations CodeQL will use them in preference to the QL source you have modified.