Uploading CodeQL analysis results to GitHub

In this article

You can use the CodeQL CLI to upload CodeQL analysis results to GitHub.

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 SARIF output

GitHub creates code scanning alerts in a repository using information from Static Analysis Results Interchange Format (SARIF) files. SARIF is designed to represent the output of a broad range of static analysis tools, and there are many features in the SARIF specification that are considered "optional". The results must use SARIF version 2.1.0. For more information, see "SARIF support for code scanning."

After analyzing a CodeQL database using the CodeQL CLI, you will have a SARIF file that contains the results. For more information, see "Analyzing your code with CodeQL queries." You can then use the CodeQL CLI to upload results to GitHub.

If you used a method other than the CodeQL CLI to generate results, you can use other upload methods. For more information, see "Uploading a SARIF file to GitHub."

Note: Uploading SARIF data to display as code scanning results in GitHub is supported for organization-owned repositories with GitHub Advanced Security enabled, and public repositories on GitHub.com. For more information, see "Managing security and analysis settings for your repository."

Generating a token for authentication with GitHub

Before you can upload your results to GitHub, you will first need to generate a personal access token with the `security_events` write permission. For more information, see "Managing your personal access tokens."

If you have installed the CodeQL CLI in a third-party CI system to create results to display in GitHub as code scanning alerts, you can use a GitHub App or personal access token to upload results to GitHub. For more information, see "Using code scanning with your

existing CI system."

Uploading results to GitHub

You can check that the SARIF properties have the supported size for upload and that the file is compatible with code scanning. For more information, see "SARIF support for code scanning".

Before you can upload results to GitHub, you must determine the best way to pass the GitHub App or personal access token you created in the previous section to the CodeQL CLI. We recommend that you review your CI system's guidance on the secure use of a secret store. The CodeQL CLI supports:

- Interfacing with a secret store using the `--github-auth-stdin` option (recommended).
- Saving the secret in the environment variable `GITHUB_TOKEN` and running the CLI without including the `--github-auth-stdin` option.
- For testing purposes you can pass the `--github-auth-stdin` command-line option and supply a temporary token via standard input.

When you have decided on the most secure and reliable method for your configuration, run `codeql github upload-results` on each SARIF results file and include `--github-auth-stdin` unless the token is available in the environment variable `GITHUB_TOKEN`.

```
# GitHub App or personal access token available from a secret store
<call-to-retrieve-secret> | codeql github upload-results \
    --repository=<repository-name> \
    --ref=<ref> --commit=<commit> \
    --sarif=<file> --github-auth-stdin

# GitHub App or personal access token available in GITHUB_TOKEN
codeql github upload-results \
    --repository=<repository-name> \
    --ref=<ref> --commit=<commit> \
    --sarif=<file>
```

Option	Required	Usage
`--repository`	✓	Specify the *OWNER/NAME* of the repository to upload data to. The owner must be an organization within an enterprise that has a license for GitHub Advanced Security and GitHub Advanced Security must be enabled for the repository, unless the repository is public. For more information, see "Managing security and analysis settings for your repository."
`--ref`	✓	Specify the name of the `ref` you checked out and analyzed so that the results can be matched to the correct code. For a branch use: `refs/heads/BRANCH-NAME`, for the head commit of a pull request use `refs/pull/NUMBER/head`, or for the GitHub-generated merge

		commit of a pull request use `refs/pull/NUMBER/merge`.
`--commit`	✓	Specify the full SHA of the commit you analyzed.
`--sarif`	✓	Specify the SARIF file to load.
`--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.

For more information, see "github upload-results."

Note: If you analyzed more than one CodeQL database for a single commit, you must have specified a SARIF category for each set of results generated by this command. When you upload the results to GitHub, code scanning uses this category to store the results for each language separately. If you forget to do this, each upload overwrites the previous results. For more information, see "Analyzing your code with CodeQL queries."

Basic example of uploading results to GitHub

The following example uploads results from the SARIF file `temp/example-repo-js.sarif` to the repository `my-org/example-repo`. It tells the code scanning API that the results are for the commit `deb275d2d5fe9a522a0b7bd8b6b6a1c939552718` on the `main` branch. The example assumes that the GitHub App or personal access token created for authentication with GitHub's REST API uses the `GITHUB_TOKEN` environment variable.

```
codeql github upload-results \
    --repository=my-org/example-repo \
    --ref=refs/heads/main --commit=deb275d2d5fe9a522a0b7bd8b6b6a1c939552718 \
    --sarif=/temp/example-repo-js.sarif
```

There is no output from this command unless the upload was unsuccessful. The command prompt returns when the upload is complete and data processing has begun. On smaller codebases, you should be able to explore the code scanning alerts in GitHub shortly afterward. You can see alerts directly in the pull request or on the **Security** tab for branches, depending on the code you checked out. For more information, see "Triaging code scanning alerts in pull requests" and "Managing code scanning alerts for your repository."

Uploading diagnostic information to GitHub if the analysis fails

When CodeQL CLI finishes analyzing a database successfully, it gathers diagnostic information such as file coverage, warnings, and errors, and includes it in the SARIF file with the results. When you upload the SARIF file to GitHub the diagnostic information is displayed on the code scanning tool status page for the repository to make it easy to see how well CodeQL is working and debug any problems. For more information, see "About the tool status page for code scanning."

However, if `codeql database analyze` fails for any reason there is no SARIF file to upload to GitHub and no diagnostic information to show on the code scanning tool status page

for the repository. This makes it difficult for users to troubleshoot analysis unless they have access to log files in your CI system.

We recommend that you configure your CI workflow to export and upload diagnostic information to GitHub when an analysis fails. You can do this using the following simple commands to export diagnostic information and upload it to GitHub.

Exporting diagnostic information if the analysis fails

You can create a SARIF file for the failed analysis using "database export-diagnostics", for example:

```
$ codeql database export-diagnostics codeql-dbs/example-repo \
    --sarif-category=javascript-typescript --format=sarif-latest \
    --output=/temp/example-repo-js.sarif
```

This SARIF file will contain diagnostic information for the failed analysis, including any file coverage information, warnings, and errors generated during the analysis.

Uploading diagnostic information if the analysis fails

You can make this diagnostic information available on the tool status page by uploading the SARIF file to GitHub using "github upload-results", for example:

```
codeql github upload-results \
    --repository=my-org/example-repo \
    --ref=refs/heads/main --commit=deb275d2d5fe9a522a0b7bd8b6b6a1c939552718 \
    --sarif=/temp/example-repo-js.sarif
```

This is the same as the process for uploading SARIF files from successful analyses.