Uploading a SARIF file to GitHub

In this article

You can upload SARIF files generated outside GitHub and see code scanning alerts from third-party tools in your repository.

> **Who can use this feature**
> People with write permissions to a repository can upload code scanning data generated outside GitHub.

Code scanning is available for all public repositories on GitHub.com. Code scanning is also available for private repositories owned by organizations that use GitHub Enterprise Cloud and have a license for GitHub Advanced Security. For more information, see "About GitHub Advanced Security."

About SARIF file uploads for code scanning

GitHub creates code scanning alerts in a repository using information from Static Analysis Results Interchange Format (SARIF) files. SARIF files can be uploaded to a repository using the API or GitHub Actions. For more information, see "Managing code scanning alerts for your repository."

You can generate SARIF files using many static analysis security testing tools, including CodeQL. The results must use SARIF version 2.1.0. For more information, see "SARIF support for code scanning."

You can upload the results using GitHub Actions, the code scanning API, or the CodeQL CLI. The best upload method will depend on how you generate the SARIF file, for example, if you use:

- GitHub Actions to run the CodeQL action, there is no further action required. The CodeQL action uploads the SARIF file automatically when it completes analysis.
- GitHub Actions to run a SARIF-compatible analysis tool, you could update the workflow to include a final step that uploads the results (see below).
- The CodeQL CLI to run code scanning in your CI system, you can use the CLI to upload results to GitHub (for more information, see "Using code scanning with your existing CI system").
- A tool that generates results as an artifact outside of your repository, you can use the code scanning API to upload the file (for more information, see "Code Scanning").

Note: For private and internal repositories, code scanning is available when GitHub Advanced Security features are enabled for the repository. If you see the error `Advanced Security must be enabled for this repository to use code scanning`, check that GitHub Advanced Security is enabled. For more information, see "Managing security and analysis settings for your repository."

Uploading a code scanning analysis with GitHub Actions

To use GitHub Actions to upload a third-party SARIF file to a repository, you'll need a workflow. For more information, see "Learn GitHub Actions."

Your workflow will need to use the `upload-sarif` action, which is part of the `github/codeql-action` repository. It has input parameters that you can use to configure the upload. The main input parameters you'll use are:

- `sarif-file`, which configures the file or directory of SARIF files to be uploaded. The directory or file path is relative to the root of the repository.
- `category` (optional), which assigns a category for results in the SARIF file. This enables you to analyze the same commit in multiple ways and review the results using the code scanning views in GitHub. For example, you can analyze using multiple tools, and in mono-repos, you can analyze different slices of the repository based on the subset of changed files.

For more information see the `upload-sarif` action.

The `upload-sarif` action can be configured to run when the `push` and `scheduled` event occur. For more information about GitHub Actions events, see "Events that trigger workflows."

If your SARIF file doesn't include `partialFingerprints`, the `upload-sarif` action will calculate the `partialFingerprints` field for you and attempt to prevent duplicate alerts. GitHub can only create `partialFingerprints` when the repository contains both the SARIF file and the source code used in the static analysis. For more information about preventing duplicate alerts, see "SARIF support for code scanning."

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".

Example workflow for SARIF files generated outside of a repository

You can create a new workflow that uploads SARIF files after you commit them to your repository. This is useful when the SARIF file is generated as an artifact outside of your repository.

This example workflow runs anytime commits are pushed to the repository. The action uses the `partialFingerprints` property to determine if changes have occurred. In addition to running when commits are pushed, the workflow is scheduled to run once per week. For more information, see "Events that trigger workflows."

This workflow uploads the `results.sarif` file located in the root of the repository. For more information about creating a workflow file, see "Learn GitHub Actions."

Alternatively, you could modify this workflow to upload a directory of SARIF files. For example, you could place all SARIF files in a directory in the root of your repository called `sarif-output` and set the action's input parameter `sarif_file` to `sarif-output`. Note that if you upload a directory, each SARIF file must include a unique `runAutomationDetails.id` to define the category for the results. For more information, see "SARIF support for code scanning."

```yaml
name: "Upload SARIF"

# Run workflow each time code is pushed to your repository and on a schedule.
# The scheduled workflow runs every Thursday at 15:45 UTC.
on:
```

```
  push:
  schedule:
    - cron: '45 15 * * 4'

jobs:
  build:
    runs-on: ubuntu-latest
    permissions:
      # required for all workflows
      security-events: write
      # only required for workflows in private repositories
      actions: read
      contents: read
    steps:
      # This step checks out a copy of your repository.
      - name: Checkout repository
        uses: actions/checkout@v4
      - name: Upload SARIF file
        uses: github/codeql-action/upload-sarif@v2
        with:
          # Path to SARIF file relative to the root of the repository
          sarif_file: results.sarif
          # Optional category for the results
          # Used to differentiate multiple results for one commit
          category: my-analysis-tool
```

Example workflow that runs the ESLint analysis tool

If you generate your third-party SARIF file as part of a continuous integration (CI) workflow, you can add the `upload-sarif` action as a step after running your CI tests. If you don't already have a CI workflow, you can create one using a GitHub Actions template. For more information, see the "Quickstart for GitHub Actions."

This example workflow runs anytime commits are pushed to the repository. The action uses the `partialFingerprints` property to determine if changes have occurred. In addition to running when commits are pushed, the workflow is scheduled to run once per week. For more information, see "Events that trigger workflows."

The workflow shows an example of running the ESLint static analysis tool as a step in a workflow. The `Run ESLint` step runs the ESLint tool and outputs the `results.sarif` file. The workflow then uploads the `results.sarif` file to GitHub using the `upload-sarif` action. For more information about creating a workflow file, see "Understanding GitHub Actions."

```
name: "ESLint analysis"

# Run workflow each time code is pushed to your repository and on a schedule.
# The scheduled workflow runs every Wednesday at 15:45 UTC.
on:
  push:
  schedule:
    - cron: '45 15 * * 3'

jobs:
  build:
    runs-on: ubuntu-latest
    permissions:
      # required for all workflows
      security-events: write
      # only required for workflows in private repositories
      actions: read
      contents: read
    steps:
      - uses: actions/checkout@v4
      - name: Run npm install
        run: npm install
```

```
      # Runs the ESlint code analysis
      - name: Run ESLint
        # eslint exits 1 if it finds anything to report
        run: node_modules/.bin/eslint build docs lib script spec-main -f
node_modules/@microsoft/eslint-formatter-sarif/sarif.js -o results.sarif || true
      # Uploads results.sarif to GitHub repository using the upload-sarif action
      - uses: github/codeql-action/upload-sarif@v2
        with:
          # Path to SARIF file relative to the root of the repository
          sarif_file: results.sarif
```

Further reading

- "Troubleshooting SARIF uploads"
- "Workflow syntax for GitHub Actions"
- "Viewing workflow run history"
- "Using code scanning with your existing CI system"
- "Code Scanning"