Error: "No source code was seen during the build" or "The process '/opt/hostedtoolcache/CodeQL/0.0.0-20200630/x64/codeql/codeql' failed with exit code 32"

When CodeQL fails to find any source code, you need to resolve this problem to unblock code scanning analysis.

If your workflow fails with `Error: "No source code was seen during the build"` or `The process '/opt/hostedtoolcache/CodeQL/0.0.0-20200630/x64/codeql/codeql' failed with exit code 32`, this indicates that CodeQL was unable to monitor your code. There are six possible reasons for this:

1. The repository may not contain source code that is written in languages supported by CodeQL. Check the list of supported languages and, if this is the case, remove the CodeQL workflow. For more information, see "About code scanning with CodeQL."

2. Automatic language detection identified a supported language, but there is no analyzable code of that language in the repository. A typical example is when our language detection service finds a file associated with a particular programming language like a `.h`, or `.gyp` file, but no corresponding executable code is present in the repository. To solve the problem, you can manually define the languages you want to analyze by updating the list of languages in the `language` matrix. For example, the following configuration will analyze only Go, and JavaScript.

   ```
   strategy:
     fail-fast: false
     matrix:
       # Override automatic language detection by changing the list below.
       # Supported options are listed in a comment in the default workflow.
       language: ['go', 'javascript-typescript']
   ```

   For more information, see the workflow extract in "Some languages were not analyzed with CodeQL advanced setup".

3. Your code scanning workflow is analyzing a compiled language (C, C++, C#, Go, or Java), but the code was not compiled. By default, the CodeQL analysis workflow contains an `autobuild` step, however, this step represents a best effort process, and may not succeed in building your code, depending on your specific build environment. Compilation may also fail if you have removed the `autobuild` step and did not include build steps manually. For more information about specifying build steps, see "CodeQL code scanning for compiled languages."

4. Your workflow is analyzing a compiled language (C, C++, C#, Go, or Java), but portions of your build are cached to improve performance (most likely to occur with build systems like Gradle or Bazel). Since CodeQL observes the activity of the

compiler to understand the data flows in a repository, CodeQL requires a complete build to take place in order to perform analysis.

5. Your workflow is analyzing a compiled language (C, C++, C#, Go, or Java), but compilation does not occur between the `init` and `analyze` steps in the workflow. CodeQL requires that your build happens in between these two steps in order to observe the activity of the compiler and perform analysis.

6. Your compiled code (in C, C++, C#, Go, or Java) was compiled successfully, but CodeQL was unable to detect the compiler invocations. The most common causes are:

    - Running your build process in a separate container to CodeQL. For more information, see "Running CodeQL code scanning in a container."
    - Building using a distributed build system external to GitHub Actions, using a daemon process.
    - CodeQL isn't aware of the specific compiler you are using.

If you encounter another problem with your specific compiler or configuration, contact us through the GitHub Support portal.

For more information about specifying build steps, see "CodeQL code scanning for compiled languages."