

RFC 107 IMPLEMENTED: Overlapping LSIF roots

Editor: eric@sourcegraph.com

Status: WIP

Approvals: Chris, Michael

Background

The LSIF server currently accepts uploads along with a *root*, which is a file path relative to the project base in which all indexed files are located. This is meant to aid in multi-language and large monorepo indexing. We had originally planned to use glob patterns to support multiple languages, but went with a simplified model to implement as it seemed good enough at the time. See [RFC 24](#) for additional context on the original feature planning.

Problem

Let's use the configuration of [sourcegraph/sourcegraph](#) as an example. We (would) currently index the following roots:

- cmd
- internal
- enterprise/web
- enterprise/cmd
- enterprise/internal
- web
- shared
- browser
- lsif

The web, shared, browser, lsif, and enterprise/web projects are all distinct (they have a unique package.json and tsconfig, although they do extend a common configuration in the root). It makes sense that they would index separately. However, the cmd, internal, enterprise/cmd, and enterprise/internal directories all share a common go.mod/go.sum in the root directory. These logically form a single project and should be indexed together. This is not possible with our current strategy of requiring index roots to be disjoint for a single commit: we cannot have a go index root of / and typescript index roots within it.

This proposal is being announced now as it will greatly simplify the logic required by [RFC 101](#), which will produce our automatic LSIF configuration algorithms.

Proposal

The code which handles selecting dumps based on roots are confined to three functions in the lsif-server:

- [findClosestDump](#): given a repo, commit, and file path, the dump with a prefix of that file path is returned with the minimum commit distance in the same repo
- [updateDumpsVisibleFromTip](#): given a repo and the commit which is considered the *tip* of the default branch, update the *visibleAtTip* flag for all dumps of that repo; a dump is visible if there is no dump with a smaller commit distance to the tip with a more general root (a prefix of the given root)
- [deleteOverlappingDumps](#): given a repo, commit, and root, delete all dumps with a less specific root (prefixed by the given root) at the same commit

We need to alter the *root* of each LSIF upload with some additional data. We had originally planned on pulling all of the unique file extensions from the documents in the dump and storing them as a sorted list in the table. This has some issues with files that are indexed by multiple indexers (for example, header files for c, c++, objc, etc).

Instead, I suggest that we pull the *tool name* from the metadata node in the first line of the upload. We can then treat uploads from each tool the same way we treated all dumps before: a more general root will block the visibility of a less general root. If you upload with indexer A an upload at root */web* and then upload with the same indexer an upload at root */* for the same commit, the previous one will be deleted. Doing the same but at a later commit will still shadow the earlier upload.

Uploads from two different indexers do not have these rules: we will be able to upload data rooted at */* from indexer A and data rooted at */web* from indexer B. These will not shadow each other. When asking for the closest dump, we will return the *set* of dumps that are visible from the current commit that contain that file. This is equivalent to running the existing query for each possible indexer (but can be done in a more efficient way in practice without knowing the list of all indexers).

Note that using the tool name will have a nice property that we don't get from knowing file paths alone. On the deletion of a file, the indexer will remain the same, but if we were to use file paths, then the old dumps will still be visible as they have a member of the set that's not in a later dump. This would mean that deleted files are always reachable within the traversal limit, depending on implementation.

This requires changing the idea of a closest dump into an idea of a set of closest dumps. For a first approximation, we should just keep the first such dump we see, but as a longer goal we should query multiple dumps in case reading one does not give complete results.

Definition of success

We can successfully upload, maintain the visibility of, and query non-disjoint roots in the same commit without a regression of the current tests. New tests for the querying and visibility of roots are written and run successfully.