

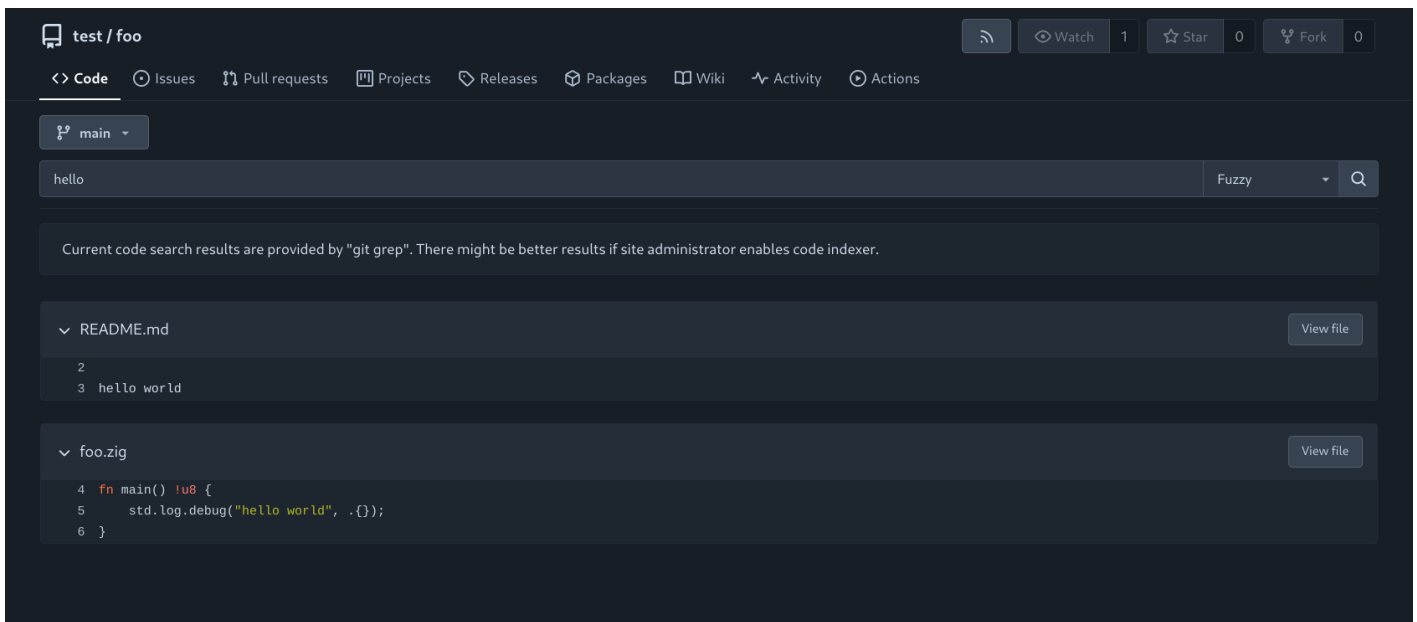
Search

- [Code Search](#)
- [Issue Search](#)

Code Search

Forgejo supports code search through an indexer and `git-grep` as a fallback when `REPO_INDEXER_ENABLED` is disabled.

Basic (git-grep)



When `REPO_INDEXER_ENABLED` is set to `false`, code search is restricted to a single repository, utilizing the `git-grep` command.

Supported Options

The following options are currently available for code search while using `git-grep`.

- **Exact:** Perform an exact match on the provided expression.
- **Union:** Conduct a union match, returning results that contain at least one of the specified keywords. For example, a search query containing `hello world` will yield results with either `hello` or `world`.
- **RegExp:** Utilize the provided regular expression to perform a pattern-based match (matches will not be highlighted).

Scope

Since `git-grep` is performed on the fly, it can be executed on any valid branch or tag. The currently active branch/tag is displayed as the default value in the dropdown menu above the search bar, allowing users to easily switch between branches and tags.

Searching within a specific directory (or file) executes `git-grep` using a [literal pathspec](#) for the given path. If `REPO_INDEXER_INCLUDE` has been set by the administrator, the filter is added if and only if it matches one of the globs.

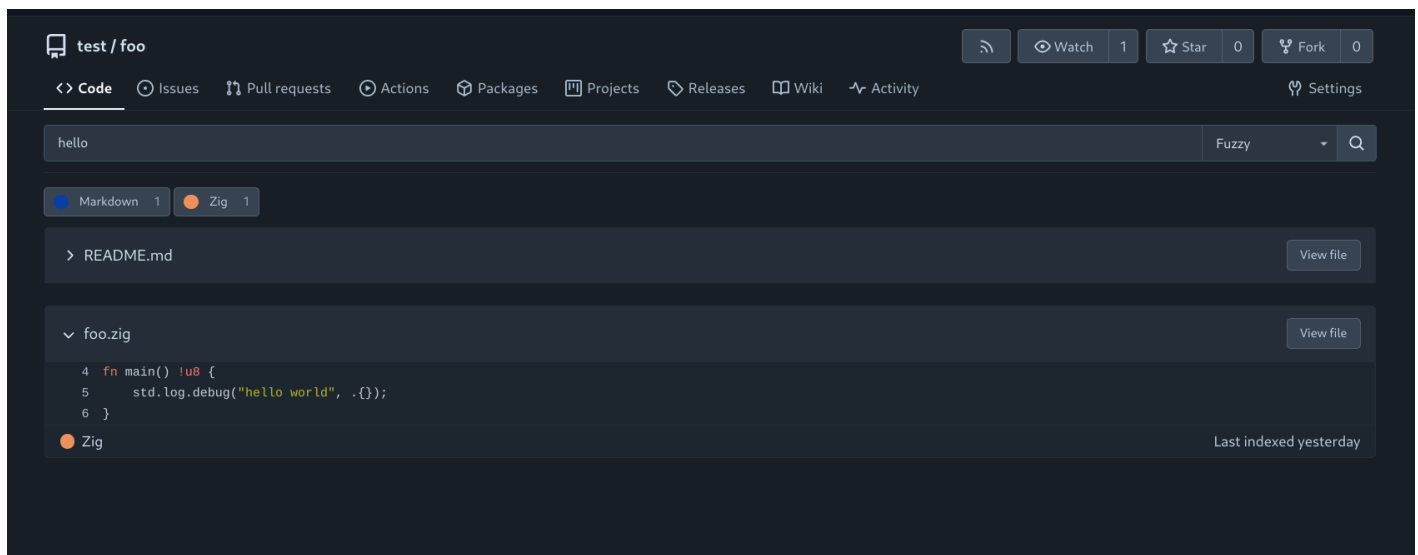
Example

Performing a search for `foo` at `/{{user}}/{{repo}}/src/branch/main?path=src` returns results that belong to the branch `main` inside the directory `/src`.

```
main
├─ docs -> [...]
└─ src
    └─ main.go
    └─ utils.go
```

In the above figure, the search would match results for `foo` in `main.go` and `utils.go`, but not from `docs/*`.

Indexer



For complex searches or cross-repository queries across an entire organization or instance, `REPO_INDEXER_ENABLED` must be set to `true`. This enables code search via the selected indexer ([REPO_INDEXER_TYPE](#)).

Supported Options

The following options are currently available for code search while using an indexer.

- **Exact:** Perform an exact match on the provided expression.
- **Union:** Conduct a union match, returning results that contain at least one of the specified keywords. For example, a search query containing `hello world` will yield results with either `hello` or `world`.
- **Fuzzy** (if enabled by the administrator): Conduct a fuzzy search, returning results that contain the keyword within a maximum edit distance of 2. For example, a search query containing `hello` will yield results with:
 - **edit distance of 0:** `hello`
 - **edit distance of 1:** For example, `hlllo` (delete), `helloo` (add), `hallo` (modify).

“ [!note] Note Fuzzy search requires the admin to set `REPO_INDEXER_FUZZY_ENABLED` .

Scope

Please note that when using the repository indexer, search results are limited to the contents of the default branch of each repository.

Similar to basic search, searching within a directory (or file) is also possible for advanced search. However, unlike basic search, the search is more granular as it applies the filter but selectively includes/excludes files depending on `REPO_INDEXER_INCLUDE` / `REPO_INDEXER_EXCLUDE` .

Issue Search

Since v11, Issue search on Forgejo replaced its previously dropdown-defined behavior of searches with a query syntax defined by operators.

Query Syntax

Keywords

1. Terms

Performs a fuzzy match search of each term in the query.

For example,

- `foo` will return results that contain `foo`, or any of its derivatives (such as `for` or `food`)
- `foo bar` will return results that contain `foo` and/or `bar` or any of their derivatives

2. Phrase

Performs an exact match for the provided phrase. This is done by wrapping the phrase in double quotes.

For example,

- `"foo"` will return results that contain an exact match of `foo`
- `"foo bar"` will return results that contain an exact match of `foo bar`

Operators

1. Negation

Negates the keyword by returning results that do not contain the keyword. This is done by prefixing the keyword with a minus symbol.

For example,

- `-foo` will return results that do not contain `foo` or its derivatives

- `-"foo bar"` will return results that do not contain `foo bar`

2. Required

Denotes that the provided keyword **MUST** be present. This is done by prefixing the keyword with a plus symbol.

For example,

- `+foo +bar` will return results that contain both `foo` and `bar` (or its fuzzy derivatives)
- `+"foo" +"bar"` will return results that contain exactly both `foo` and `bar`
- `+"foo bar" +"baz"` will return results that contain exactly both `foo bar` and `baz`

Escaping

All operators may be escaped by adding a backslash in front of the operators.

For example,

- `"\\\""` will search for `"`
- `\\+1` will search for `+1`

Filters

Filters listed below may be escaped by using a phrase search. For example, `"is:open"`.

State

These filters are mutually exclusive. If multiple state filters are used, then the rightmost filter takes precedence.

Filter	Description
<code>is:open</code> or <code>-is:closed</code>	Filters issues/pull requests that are open
<code>is:closed</code> or <code>-is:open</code>	Filters issues/pull requests that are closed
<code>is:all</code>	Filters issues/pull requests that are either open or closed

Examples,

- `is:open +"bug"`: Searches for `bug` among open issues/PRs.
- `is:all -"backport"`: Search for all issues/PRs that do not contain the term "backport".

- `"is:all"`: Literal search for `is:all`. Matches Issues/PRs that contain the term `is:all`.

User

These filter issues/pull requests by a username.

Filter	Description
<code>author:*</code>	Authored By
<code>assignee:*</code>	Assigned To
<code>review:*</code>	Reviewed By
<code>mentions:*</code>	Mentions the user

Examples,

- `author:forgejo is:all`: All issues/PRs created by user `forgejo`.

Misc.

Filter	Description
<code>sort:<by>:[asc/desc]</code>	Sorts issues/PRs. Where, <code><by></code> is one of <code>created</code> , <code>comments</code> , <code>updated</code> , <code>deadline</code>
<code>modified:[>/<]<date></code>	Last updated date in ISO format

Examples,

- `sort:created:desc is:all`: Sort by the latest issues/PRs.
- `sort:comments:desc`: Sort by the most commented issues.
- `modified:>2025-12-05`: Searches for issues/PRs modified after `2025-12-05`.
- `modified:2025-12-05`: Searches for issues/PRs modified exactly on `2025-12-05`.
- `modified:<2025-12-05`: Searches for issues/PRs modified before `2025-12-05`.