In this article

Ranking search results
Rate limit
Constructing a search query
Limitations on query length
Timeouts and incomplete results
Access errors or missing search results
Search code
Search commits
Search issues and pull requests
Search labels
Search repositories
Search topics
Search users
Text match metadata

Search

The GitHub Search API lets you to search for the specific item efficiently.

The Search API helps you search for the specific item you want to find. For example, you can find a user or a specific file in a repository. Think of it the way you think of performing a search on Google. It's designed to help you find the one result you're looking for (or maybe the few results you're looking for). Just like searching on Google, you sometimes want to see a few pages of search results so that you can find the item that best meets your needs. To satisfy that need, the GitHub Search API provides **up to 1,000 results for each search**

You can narrow your search using queries. To learn more about the search query syntax, see "Constructing a search query."

Ranking search results

Unless another sort option is provided as a query parameter, results are sorted by best match in descending order. Multiple factors are combined to boost the most relevant item to the top of the result list.

Rate limit

The Search API has a custom rate limit. For requests using Basic Authentication, OAuth, or client ID and secret, you can make up to 30 requests per minute. For unauthenticated requests, the rate limit allows you to make up to 10 requests per minute.

See the rate limit documentation for details on determining your current rate limit status.

Constructing a search query

Each endpoint in the Search API uses query parameters to perform searches on GitHub. See the individual endpoint in the Search API for an example that includes the endpoint and query parameters.

A query can contain any combination of search qualifiers supported on GitHub. The format of the search query is:

```
SEARCH_KEYWORD_1 SEARCH_KEYWORD_N QUALIFIER_1 QUALIFIER_N
```

For example, if you wanted to search for all *repositories* owned by `defunkt` that contained the word `GitHub` and `Octocat` in the README file, you would use the following query with the *search repositories* endpoint:

```
GitHub Octocat in:readme user:defunkt
```

Note: Be sure to use your language's preferred HTML-encoder to construct your query strings. For example:

```
// JavaScript
const queryString = 'q=' + encodeURIComponent('GitHub Octocat in:readme user:defunkt
```

See "Searching on GitHub" for a complete list of available qualifiers, their format, and an example of how to use them. For information about how to use operators to match specific quantities, dates, or to exclude results, see "Understanding the search syntax."

Limitations on query length

The Search API does not support queries that:

- are longer than 256 characters (not including operators or qualifiers).
- have more than five `AND`, `OR`, or `NOT` operators.

These search queries will return a "Validation failed" error message.

Timeouts and incomplete results

To keep the Search API fast for everyone, we limit how long any individual query can run. For queries that exceed the time limit, the API returns the matches that were already found prior to the timeout, and the response has the `incomplete_results` property set to `true`.

Reaching a timeout does not necessarily mean that search results are incomplete. More results might have been found, but also might not.

Access errors or missing search results

You need to successfully authenticate and have access to the repositories in your search queries, otherwise, you'll see a `422 Unprocessible Entry` error with a "Validation Failed" message. For example, your search will fail if your query includes `repo:`, `user:`, or `org:` qualifiers that request resources that you don't have access to when you sign in on GitHub.

When your search query requests multiple resources, the response will only contain the resources that you have access to and will **not** provide an error message listing the resources that were not returned.

For example, if your search query searches for the `octocat/test` and `codertocat/test` repositories, but you only have access to `octocat/test`, your response will show search results for `octocat/test` and nothing for `codertocat/test`. This behavior mimics how search works on GitHub.

Search code

Searches for query terms inside of a file. This method returns up to 100 resultsper page.

When searching for code, you can get text match metadata for the file**content** and file **path** fields when you pass the `text-match` media type. For more details about how to receive highlighted search results, see Text match metadata.

For example, if you want to find the definition of the `addClass` function inside jQuery repository, your query would look something like this:

```
q=addClass+in:file+language:js+repo:jquery/jquery
```

This query searches for the keyword `addClass` within a file's contents. The query limits the search to files where the language is JavaScript in the `jquery/jquery` repository.

Considerations for code search

Due to the complexity of searching code, there are a few restrictions on how searches are performed:

- Only the *default branch* is considered. In most cases, this will be the `master` branch.
- Only files smaller than 384 KB are searchable.
- You must always include at least one search term when searching source code. For example, searching for `language:go` is not valid, while `amazing language:go` is.

```
GET /search/code
```

Parameters

Name	Type	In	Description
`accept`	string	header	Setting to `application/vnd.github.v3+json` is recommended.
`q`	string	query	The query contains one or more search keywords and qualifiers. Qualifiers allow you to limit your search to specific areas of GitHub. The REST API supports the same qualifiers as GitHub.com. To learn more about the format of the query, see Constructing a search query. See "Searching code" for a detailed list of qualifiers.

Name	Type	In	Description
`sort`	string	query	Sorts the results of your query. Can only be `indexed`, which indicates how recently a file has been indexed by the GitHub search infrastructure. Default: best match
`order`	string	query	Determines whether the first search result returned is the highest number of matches (`desc`) or lowest number of matches (`asc`). This parameter is ignored unless you provide `sort`. Default: `desc`
`per_page`	integer	query	Results per page (max 100) Default: `30`
`page`	integer	query	Page number of the results to fetch. Default: `1`

Code samples

Shell

```
curl \
  -H "Accept: application/vnd.github.v3+json" \
  https://api.github.com/search/code
```

JavaScript (@octokit/core.js)

```
await octokit.request('GET /search/code', {
  q: 'q'
})
```

Response

```
Status: 200 OK
```

```
{
  "total_count": 7,
  "incomplete_results": false,
  "items": [
    {
      "name": "classes.js",
      "path": "src/attributes/classes.js",
      "sha": "d7212f9dee2dcc18f084d7df8f417b80846ded5a",
      "url": "https://api.github.com/repositories/167174/contents/src/attributes/c
      "git_url": "https://api.github.com/repositories/167174/git/blobs/d7212f9dee2
      "html_url": "https://github.com/jquery/jquery/blob/825ac3773694e0cd23ee74895
      "repository": {
        "id": 167174,
        "node_id": "MDEwOlJlcG9zaXRvcnkxNjcxNzQ=",
        "name": "jquery",
        "full_name": "jquery/jquery",
        "owner": {
          "login": "jquery",
          "id": 70142,
          "node_id": "MDQ6VXNlcjcwMTQy",
          "avatar_url": "https://0.gravatar.com/avatar/6906f317a4733f4379b06c32229
          "gravatar_id": "",
          "url": "https://api.github.com/users/jquery",
          "html_url": "https://github.com/jquery"
```


Not modified

```
Status: 304 Not Modified
```

Forbidden

```
Status: 403 Forbidden
```

Validation failed

```
Status: 422 Unprocessable Entity
```

Service unavailable

```
Status: 503 Service Unavailable
```

Notes

- Works with GitHub Apps

Search commits

Find commits via various criteria on the default branch (usually `master`). This method returns up to

100 results per page.

When searching for commits, you can get text match metadata for the **message** field when you provide the `text-match` media type. For more details about how to receive highlighted search results, see Text match metadata.

For example, if you want to find commits related to CSS in the octocat/Spoon-Knife repository. Your query would look something like this:

```
q=repo:octocat/Spoon-Knife+css
```

```
GET /search/commits
```

Parameters

Name	Type	In	Description
`accept`	string	header	Setting to `application/vnd.github.v3+json` is recommended.
`q`	string	query	The query contains one or more search keywords and qualifiers. Qualifiers allow you to limit your search to specific areas of GitHub. The REST API supports the same qualifiers as GitHub.com. To learn more about the format of the query, see Constructing a search query. See "Searching commits" for a detailed list of qualifiers.
`sort`	string	query	Sorts the results of your query by `author-date` or `committer-date`. Default: best match
`order`	string	query	Determines whether the first search result returned is the highest number of matches (`desc`) or lowest number of matches (`asc`). This parameter is ignored unless you provide `sort`. Default: `desc`
`per_page`	integer	query	Results per page (max 100) Default: `30`
`page`	integer	query	Page number of the results to fetch. Default: `1`

Code samples

Shell

```
curl \
  -H "Accept: application/vnd.github.v3+json" \
  https://api.github.com/search/commits
```

JavaScript (@octokit/core.js)

```
await octokit.request('GET /search/commits', {
  q: 'q'
})
```

Response

```
Status: 200 OK
```


```
{
  "total_count": 1,
  "incomplete_results": false,
  "items": [
    {
      "url": "https://api.github.com/repos/octocat/Spoon-Knife/commits/bb4cc8d3b2e
      "sha": "bb4cc8d3b2e14b3af5df699876dd4ff3acd00b7f",
      "html_url": "https://github.com/octocat/Spoon-Knife/commit/bb4cc8d3b2e14b3af
      "comments_url": "https://api.github.com/repos/octocat/Spoon-Knife/commits/bb
      "commit": {
        "url": "https://api.github.com/repos/octocat/Spoon-Knife/git/commits/bb4cc
        "author": {
          "date": "2014-02-04T14:38:36-08:00",
          "name": "The Octocat",
          "email": "octocat@nowhere.com"
        },
        "committer": {
          "date": "2014-02-12T15:18:55-08:00",
          "name": "The Octocat",
          "email": "octocat@nowhere.com"
        },
        "message": "Create styles.css and updated README",
        "tree": {
          "url": "https://api.github.com/repos/octocat/Spoon-Knife/git/trees/a639c
```


Not modified

```
Status: 304 Not Modified
```

Notes

- Works with GitHub Apps

Search issues and pull requests

Find issues by state and keyword. This method returns up to 100 resultsper page.

When searching for issues, you can get text match metadata for the issue**title**, issue **body**, and issue **comment body** fields when you pass the `text-match` media type. For more details about how to receive highlighted search results, see Text match metadata.

For example, if you want to find the oldest unresolved Python bugs on Windows. Your query might

look something like this.

```
q=windows+label:bug+language:python+state:open&sort=created&order=asc
```

This query searches for the keyword `windows`, within any open issue that is labeled as `bug`. The search runs across repositories whose primary language is Python. The results are sorted by creation date in ascending order, which means the oldest issues appear first in the search results.

Note: For user-to-server GitHub App requests, you can't retrieve a combination of issues and pull requests in a single query. Requests that don't include the `is:issue` or `is:pull-request` qualifier will receive an HTTP `422 Unprocessable Entity` response. To get results for both issues and pull requests, you must send separate queries for issues and pull requests. For more information about the `is` qualifier, see "Searching only issues or pull requests."

```
GET /search/issues
```

Parameters

Name	Type	In	Description
`accept`	string	header	Setting to `application/vnd.github.v3+json` is recommended.
`q`	string	query	The query contains one or more search keywords and qualifiers. Qualifiers allow you to limit your search to specific areas of GitHub. The REST API supports the same qualifiers as GitHub.com. To learn more about the format of the query, see Constructing a search query. See "Searching issues and pull requests" for a detailed list of qualifiers.
`sort`	string	query	Sorts the results of your query by the number of `comments`, `reactions`, `reactions-+1`, `reactions--1`, `reactions-smile`, `reactions-thinking_face`, `reactions-heart`, `reactions-tada`, or `interactions`. You can also sort results by how recently the items were `created` or `updated`, Default: best match
`order`	string	query	Determines whether the first search result returned is the highest number of matches (`desc`) or lowest number of matches (`asc`). This parameter is ignored unless you provide `sort`. Default: `desc`
`per_page`	integer	query	Results per page (max 100) Default: `30`
`page`	integer	query	Page number of the results to fetch. Default: `1`

Code samples

Shell

```
curl \
  -H "Accept: application/vnd.github.v3+json" \
  https://api.github.com/search/issues
```

JavaScript (@octokit/core.js)

```
await octokit.request('GET /search/issues', {
  q: 'q'
})
```

Response

```
Status: 200 OK
```


```
{
  "total_count": 280,
  "incomplete_results": false,
  "items": [
    {
      "url": "https://api.github.com/repos/batterseapower/pinyin-toolkit/issues/13
      "repository_url": "https://api.github.com/repos/batterseapower/pinyin-toolki
      "labels_url": "https://api.github.com/repos/batterseapower/pinyin-toolkit/is
      "comments_url": "https://api.github.com/repos/batterseapower/pinyin-toolkit/
      "events_url": "https://api.github.com/repos/batterseapower/pinyin-toolkit/is
      "html_url": "https://github.com/batterseapower/pinyin-toolkit/issues/132",
      "id": 35802,
      "node_id": "MDU6SXNzdWUzNTgwMg==",
      "number": 132,
      "title": "Line Number Indexes Beyond 20 Not Displayed",
      "user": {
        "login": "Nick3C",
        "id": 90254,
        "node_id": "MDQ6VXNlcjkwMjU0",
        "avatar_url": "https://secure.gravatar.com/avatar/934442aadfe3b2f4630510de
        "gravatar_id": "",
        "url": "https://api.github.com/users/Nick3C",
        "html_url": "https://github.com/Nick3C",
        "followers_url": "https://api.github.com/users/Nick3C/followers"
```


Not modified

```
Status: 304 Not Modified
```

Forbidden

```
Status: 403 Forbidden
```

Validation failed

```
Status: 422 Unprocessable Entity
```

Service unavailable

```
Status: 503 Service Unavailable
```

Notes

- Works with GitHub Apps

Search labels

Find labels in a repository with names or descriptions that match search keywords. Returns up to 100 results per page.

When searching for labels, you can get text match metadata for the label**name** and **description** fields when you pass the `text-match` media type. For more details about how to receive highlighted search results, see Text match metadata.

For example, if you want to find labels in the `linguist` repository that match `bug`, `defect`, or `enhancement`. Your query might look like this:

```
q=bug+defect+enhancement&repository_id=64778136
```

The labels that best match the query appear first in the search results.

```
GET /search/labels
```

Parameters

Name	Type	In	Description
`accept`	string	header	Setting to `application/vnd.github.v3+json` is recommended.
`repository_id`	integer	query	The id of the repository.
`q`	string	query	The search keywords. This endpoint does not accept qualifiers in the query. To learn more about the format of the query, see Constructing a search query.
`sort`	string	query	Sorts the results of your query by when the label was `created` or `updated`. Default: best match
`order`	string	query	Determines whether the first search result returned is the highest number of matches (`desc`) or lowest number of matches (`asc`). This parameter is ignored unless you provide `sort`. Default: `desc`
`per_page`	integer	query	Results per page (max 100) Default: `30`
`page`	integer	query	Page number of the results to fetch. Default: `1`

Code samples

Shell

```
curl \
  -H "Accept: application/vnd.github.v3+json" \
  https://api.github.com/search/labels
```

JavaScript (@octokit/core.js)

```
await octokit.request('GET /search/labels', {
  repository_id: 42,
  q: 'q'
})
```

Response

```
Status: 200 OK
```


```
{
  "total_count": 2,
  "incomplete_results": false,
  "items": [
    {
      "id": 418327088,
      "node_id": "MDU6TGFiZWw0MTgzMjcwODg=",
      "url": "https://api.github.com/repos/octocat/linguist/labels/enhancement",
      "name": "enhancement",
      "color": "84b6eb",
      "default": true,
      "description": "New feature or request.",
      "score": 1
    },
    {
      "id": 418327086,
      "node_id": "MDU6TGFiZWw0MTgzMjcwODY=",
      "url": "https://api.github.com/repos/octocat/linguist/labels/bug",
      "name": "bug",
      "color": "ee0701",
      "default": true,
      "description": "Something isn't working.",
      "score": 1
    }
  ]
```


Not modified

```
Status: 304 Not Modified
```

Forbidden

```
Status: 403 Forbidden
```

Resource not found

```
Status: 404 Not Found
```

Validation failed

```
Status: 422 Unprocessable Entity
```

Notes

- Works with GitHub Apps

Search repositories

Find repositories via various criteria. This method returns up to 100 resultsper page.

When searching for repositories, you can get text match metadata for the**name** and **description** fields when you pass the `text-match` media type. For more details about how to receive highlighted search results, see Text match metadata.

For example, if you want to search for popular Tetris repositories written in assembly code, your query might look like this:

```
q=tetris+language:assembly&sort=stars&order=desc
```

This query searches for repositories with the word `tetris` in the name, the description, or the README. The results are limited to repositories where the primary language is assembly. The results are sorted by stars in descending order, so that the most popular repositories appear first in the search results.

When you include the `mercy` preview header, you can also search for multiple topics by adding more `topic:` instances. For example, your query might look like this:

```
q=topic:ruby+topic:rails
```

```
GET /search/repositories
```

Parameters

Name	Type	In	Description
`accept`	string	header	Setting to `application/vnd.github.v3+json` is recommended.See preview notice

Name	Type	In	Description
`q`	string	query	The query contains one or more search keywords and qualifiers. Qualifiers allow you to limit your search to specific areas of GitHub. The REST API supports the same qualifiers as GitHub.com. To learn more about the format of the query, see Constructing a search query. See "Searching for repositories" for a detailed list of qualifiers.
`sort`	string	query	Sorts the results of your query by number of `stars`, `forks`, or `help-wanted-issues` or how recently the items were `updated`. Default: best match
`order`	string	query	Determines whether the first search result returned is the highest number of matches (`desc`) or lowest number of matches (`asc`). This parameter is ignored unless you provide `sort`. Default: `desc`
`per_page`	integer	query	Results per page (max 100) Default: `30`
`page`	integer	query	Page number of the results to fetch. Default: `1`

Code samples

Shell

```
curl \
  -H "Accept: application/vnd.github.v3+json" \
  https://api.github.com/search/repositories
```

JavaScript (@octokit/core.js)

```
await octokit.request('GET /search/repositories', {
  q: 'q'
})
```

Response

```
Status: 200 OK
```

```
{
  "total_count": 40,
  "incomplete_results": false,
  "items": [
    {
      "id": 3081286,
      "node_id": "MDEwOlJlcG9zaXRvcnkzMDgxMjg2",
      "name": "Tetris",
      "full_name": "dtrupenn/Tetris",
      "owner": {
        "login": "dtrupenn",
        "id": 872147,
        "node_id": "MDQ6VXNlcjg3MjE0Nw==",
        "avatar_url": "https://secure.gravatar.com/avatar/e7956084e75f239de85d3a31
        "gravatar_id": "",
        "url": "https://api.github.com/users/dtrupenn",
        "received_events_url": "https://api.github.com/users/dtrupenn/received_eve
        "type": "User",
        "html_url": "https://github.com/octocat",
        "followers_url": "https://api.github.com/users/octocat/followers",
        "following_url": "https://api.github.com/users/octocat/following{/other_us
        "gists_url": "https://api.github.com/users/octocat/gists{/gist_id}",
        "starred_url": "https://api.github.com/users/octocat/starred{/owner}{/repo
        "subscriptions_url": "https://api.github.com/users/octocat/subscriptions",
```


Not modified

```
Status: 304 Not Modified
```

Validation failed

```
Status: 422 Unprocessable Entity
```

Service unavailable

```
Status: 503 Service Unavailable
```

Notes

- Works with GitHub Apps

Preview notice

The `topics` property for repositories on GitHub is currently available for developers to preview. To view the `topics` property in calls that return repository results, you must provide a custom media type in the `Accept` header:

```
application/vnd.github.mercy-preview+json
```

Search topics

Find topics via various criteria. Results are sorted by best match. This method returns up to 100 results per page. See "Searching topics" for a detailed list of qualifiers.

When searching for topics, you can get text match metadata for the topic's **short_description**, **description**, **name**, or **display_name** field when you pass the `text-match` media type. For more details about how to receive highlighted search results, see Text match metadata.

For example, if you want to search for topics related to Ruby that are featured on https://github.com/topics. Your query might look like this:

```
q=ruby+is:featured
```

This query searches for topics with the keyword `ruby` and limits the results to find only topics that are featured. The topics that are the best match for the query appear first in the search results.

```
GET /search/topics
```

Parameters

Name	Type	In	Description
`accept`	string	header	This API is under preview and subject to change. See preview notice
`q`	string	query	The query contains one or more search keywords and qualifiers. Qualifiers allow you to limit your search to specific areas of GitHub. The REST API supports the same qualifiers as GitHub.com. To learn more about the format of the query, see Constructing a search query.
`per_page`	integer	query	Results per page (max 100) Default: `30`
`page`	integer	query	Page number of the results to fetch. Default: `1`

Code samples

Shell

```
curl \
  -H "Accept: application/vnd.github.mercy-preview+json" \
  https://api.github.com/search/topics
```

JavaScript (@octokit/core.js)

```
await octokit.request('GET /search/topics', {
  q: 'q',
  mediaType: {
    previews: [
      'mercy'
    ]
  }
})
```

Response

```
Status: 200 OK
```


```
{
  "total_count": 6,
  "incomplete_results": false,
  "items": [
    {
      "name": "ruby",
      "display_name": "Ruby",
      "short_description": "Ruby is a scripting language designed for simplified o
      "description": "Ruby was developed by Yukihiro \"Matz\" Matsumoto in 1995 wi
      "created_by": "Yukihiro Matsumoto",
      "released": "December 21, 1995",
      "created_at": "2016-11-28T22:03:59Z",
      "updated_at": "2017-10-30T18:16:32Z",
      "featured": true,
      "curated": true,
      "score": 1
    },
    {
      "name": "rails",
      "display_name": "Rails",
      "short_description": "Ruby on Rails (Rails) is a web application framework w
      "description": "Ruby on Rails (Rails) is a web application framework written
      "created_by": "David Heinemeier Hansson",
      "released": "December 13, 2005"
```


Not modified

```
Status: 304 Not Modified
```

Preview header missing

```
Status: 415 Unsupported Media Type
```

Notes

- Works with GitHub Apps

Preview notice

The `topics` property for repositories on GitHub is currently available for developers to preview. To view the `topics` property in calls that return repository results, you must provide a custom media type in the `Accept` header:

```
application/vnd.github.mercy-preview+json
```

☝This header is **required**.

Search users

Find users via various criteria. This method returns up to 100 results per page.

When searching for users, you can get text match metadata for the issue **login**, **email**, and **name** fields when you pass the `text-match` media type. For more details about highlighting search results, see Text match metadata. For more details about how to receive highlighted search results, see Text match metadata.

For example, if you're looking for a list of popular users, you might try this query:

```
q=tom+repos:%3E42+followers:%3E1000
```

This query searches for users with the name `tom`. The results are restricted to users with more than 42 repositories and over 1,000 followers.

```
GET /search/users
```

Parameters

Name	Type	In	Description
`accept`	string	header	Setting to `application/vnd.github.v3+json` is recommended.
`q`	string	query	The query contains one or more search keywords and qualifiers. Qualifiers allow you to limit your search to specific areas of GitHub. The REST API supports the same qualifiers as GitHub.com. To learn more about the format of the query, see Constructing a search query. See "Searching users" for a detailed list of qualifiers.
`sort`	string	query	Sorts the results of your query by number of `followers` or `repositories`, or when the person `joined` GitHub. Default: best match
`order`	string	query	Determines whether the first search result returned is the highest number of matches (`desc`) or lowest number of matches (`asc`). This parameter is ignored unless you provide `sort`. Default: `desc`
`per_page`	integer	query	Results per page (max 100) Default: `30`
`page`	integer	query	Page number of the results to fetch. Default: `1`

Code samples

Shell

```
curl \
  -H "Accept: application/vnd.github.v3+json" \
  https://api.github.com/search/users
```

JavaScript (@octokit/core.js)

```
await octokit.request('GET /search/users', {
  q: 'q'
})
```

Response

```
Status: 200 OK
```


```
{
  "total_count": 12,
  "incomplete_results": false,
  "items": [
    {
      "login": "mojombo",
      "id": 1,
      "node_id": "MDQ6VXNlcjE=",
      "avatar_url": "https://secure.gravatar.com/avatar/25c7c18223fb42a4c6ae1c8db6
      "gravatar_id": "",
      "url": "https://api.github.com/users/mojombo",
      "html_url": "https://github.com/mojombo",
      "followers_url": "https://api.github.com/users/mojombo/followers",
      "subscriptions_url": "https://api.github.com/users/mojombo/subscriptions",
      "organizations_url": "https://api.github.com/users/mojombo/orgs",
      "repos_url": "https://api.github.com/users/mojombo/repos",
      "received_events_url": "https://api.github.com/users/mojombo/received_events
      "type": "User",
      "score": 1,
      "following_url": "https://api.github.com/users/mojombo/following{/other_user
      "gists_url": "https://api.github.com/users/mojombo/gists{/gist_id}",
      "starred_url": "https://api.github.com/users/mojombo/starred{/owner}{/repo}"
      "events_url": "https://api.github.com/users/mojombo/events{/privacy}",
      "site_admin": true
```


Not modified

```
Status: 304 Not Modified
```

Validation failed

```
Status: 422 Unprocessable Entity
```

Service unavailable

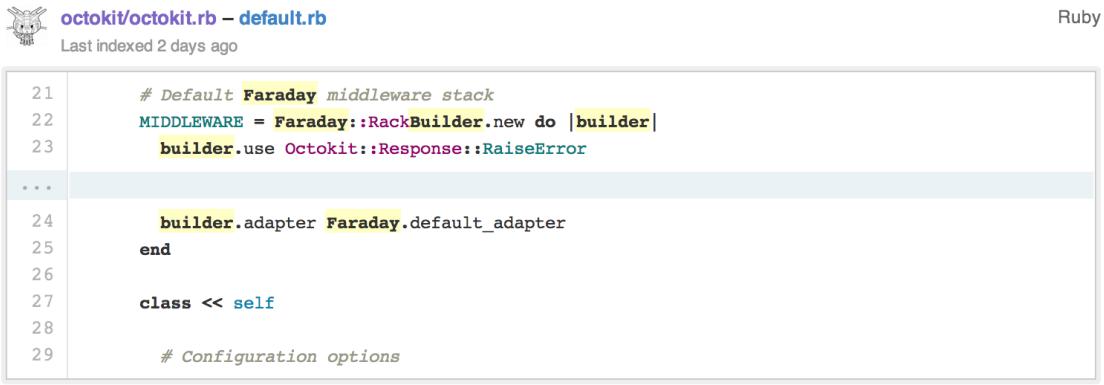
```
Status: 503 Service Unavailable
```

Notes

- Works with GitHub Apps

Text match metadata

On GitHub, you can use the context provided by code snippets and highlights in search results. The Search API offers additional metadata that allows you to highlight the matching search terms when displaying search results.



Requests can opt to receive those text fragments in the response, and every fragment is accompanied by numeric offsets identifying the exact location of each matching search term.

To get this metadata in your search results, specify the `text-match` media type in your `Accept` header.

```
application/vnd.github.v3.text-match+json
```

When you provide the `text-match` media type, you will receive an extra key in the JSON payload called `text_matches` that provides information about the position of your search terms within the text and the `property` that includes the search term. Inside the `text_matches` array, each object includes the following attributes:

Name	Description
`object_url`	The URL for the resource that contains a string property matching one of the search terms.
`object_type`	The name for the type of resource that exists at the given `object_url`.

Name	Description
`property`	The name of a property of the resource that exists at `object_url`. That property is a string that matches one of the search terms. (In the JSON returned from `object_url`, the full content for the `fragment` will be found in the property with this name.)
`fragment`	A subset of the value of `property`. This is the text fragment that matches one or more of the search terms.
`matches`	An array of one or more search terms that are present in `fragment`. The indices (i.e., "offsets") are relative to the fragment. (They are not relative to the *full* content of `property`.)

Example

Using cURL, and the example issue search above, our API request would look like this:

```
curl -H 'Accept: application/vnd.github.v3.text-match+json' \
'https://api.github.com/search/issues?q=windows+label:bug+language:python+state:open
```

The response will include a `text_matches` array for each search result. In the JSON below, we have two objects in the `text_matches` array.

The first text match occurred in the `body` property of the issue. We see a fragment of text from the issue body. The search term (`windows`) appears twice within that fragment, and we have the indices for each occurrence.

The second text match occurred in the `body` property of one of the issue's comments. We have the URL for the issue comment. And of course, we see a fragment of text from the comment body. The search term (`windows`) appears once within that fragment.

```
{
  "text_matches": [
    {
      "object_url": "https://api.github.com/repositories/215335/issues/132",
      "object_type": "Issue",
      "property": "body",
      "fragment": "comprehensive windows font I know of).\n\nIf we can find a common
      "matches": [
        {
          "text": "windows",
          "indices": [
            14,
            21
          ]
        },
        {
          "text": "windows",
          "indices": [
            78,
            85
          ]
        }
      ]
    },
    {
      "object_url": "https://api.github.com/repositories/215335/issues/comments/2568
      "object_type": "IssueComment",
      "property": "body",
      "fragment": " right after that are a bit broken IMHO :). I suppose we could ha
      "matches": [
        {
          "text": "Windows",
          "indices": [
            163,
            170
          ]
        }
      ]
    }
  ]
}
```


Or, learn how to contribute.

Still need help?

Ask the GitHub community

Contact support