Using the Explorer

You can run queries on real GitHub data using the GraphQL Explorer, an integrated development environment in your browser that includes docs, syntax highlighting, and validation errors.

In this article

About the GraphQL Explorer

GraphQL Explorer is an instance of GraphiQL, which is a "graphical interactive in-browser GraphQL IDE."

Note: GitHub has disabled mutations in the Explorer, but you can use them in your own GraphiQL instance.

Using GraphiQL

To use the GraphiQL app, download and install it from https://github.com/skevy/graphiql-app.

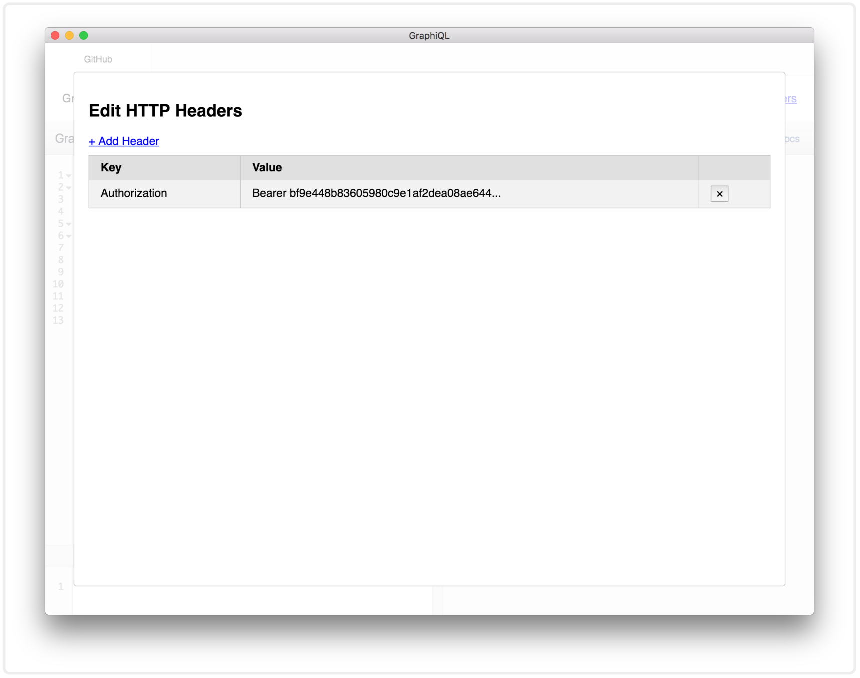
Configuring GraphiQL

Get an OAuth token.

Launch GraphiQL.

In the upper-right corner of GraphiQL, click **Edit HTTP Headers**.

In the **Key** field, enter `Authorization`. In the **Value** field, enter `Bearer <token>`, where `<token>` is your generated OAuth token.



Key	Value
Authorization	Bearer bf9e448b83605980c9e1af2dea08ae644...

Click the checkmark to the right of the token to save it.

To return to the editor, click outside of the **Edit HTTP Headers** modal.

In the **GraphQL Endpoint** field, enter `https://api.github.com/graphql`.

In the **Method** dropdown menu, select **POST**.

Note: For more information about why `POST` is the method, see "Communicating with GraphQL."

You can test your access by querying yourself:

```
query {
  viewer {
    login
  }
}
```

If everything worked correctly, this will display your login. You're all set to start making queries.

Accessing the sidebar docs

All types in a GraphQL schema include a `description` field compiled into documentation. The collapsible **Docs** pane on the right side of the Explorer page allows you to browse documentation about the type system. The docs are automatically updated and will drop deprecated fields.

The **Docs** sidebar contains the same content that is automatically generated from the schema under "Reference," though it is formatted differently in places.

Using the variable pane

Some example calls include variables written like this:

```
query($number_of_repos:Int!){
  viewer {
    name
     repositories(last: $number_of_repos) {
       nodes {
         name
       }
```

```
      }
    }
  }
  variables {
    "number_of_repos": 3
  }
```

This is the correct format to submit the call via a cURL `POST` (as long as you escape newlines).

If you want to run the call in the Explorer, enter the `query` segment in the main pane and the variables in the **Query Variables** pane below it. Omit the word `variables` from the Explorer:

```
{
   "number_of_repos": 3
}
```

Requesting support

For questions, bug reports, and discussions about GitHub Apps, OAuth Apps, and API development, explore the GitHub API Development and Support Forum. The forum is moderated and maintained by GitHub staff, but questions posted to the forum are not guaranteed to receive a reply from GitHub staff.

Consider reaching out to GitHub Support directly using the contact form for:

- guaranteed response from GitHub staff
- support requests involving sensitive data or private concerns
- feature requests
- feedback about GitHub products

Troubleshooting errors

Because GraphQL is introspective, the Explorer supports:

- Intelligent typeaheads aware of the current schema
- Validation error previews as you type

If you enter a query that is not well-formed or does not pass schema validation, a popup warns you of an error. If you run the query, the error returns in the response pane.

A GraphQL response contains several keys: a `data` hash and an `errors` array.

```
{
  "data": null,
  "errors": [
    {
      "message": "Objects must have selections (field 'nodes' returns Repository but has no selections)",
      "locations": [
        {
          "line": 5,
          "column": 8
        }
      ]
    }
  ]
}
```

It's possible you might run into an unexpected error that is not related to the schema. If this happens, the message will include a reference code you can use when reporting the issue:

```
{
  "data": null,
  "errors": [
    {
      "message": "Something went wrong while executing your query. This is most likely a GitHub bug. Please include \"7571:3FF6:552G94B:69F45B7:5
    }
  ]
}
```

Note: GitHub recommends checking for errors before using data in a production environment. In GraphQL, failure is not total: portions of GraphQL queries may succeed while others fail.