Using the API to manage Projects

In this article

You can use the GraphQL API to automate your projects.

GitHub CLI curl

This article demonstrates how to use the GraphQL API to manage a project. For more information about how to use the API in a GitHub Actions workflow, see "Automating Projects using Actions." For a full list of the available data types, see "Reference."

Authentication

In all of the following `curl` command examples, replace `TOKEN` with a token that has the `read:project` scope (for queries) or `project` scope (for queries and mutations). The token can be a personal access token (classic) for a user or an installation access token for a GitHub App. For more information about creating a personal access token, see "Managing your personal access tokens." For more information about creating an installation access token for a GitHub App, see "Generating an installation access token for a GitHub App."

When using an installation access token for a GitHub App, some GraphQL mutations require additional permissions. For example, when using the `createProjectV2` mutation, if you specify a `repositoryId` input parameter, the `Contents` permission for that repository is also required in order to link the project to the target repository.

> To learn more about GitHub CLI, see "About GitHub CLI."

Before running GitHub CLI commands, you must authenticate by running `gh auth login --scopes "project"`. If you only need to read, but not edit, projects, you can provide the `read:project` scope instead of `project`. For more information on command line authentication, see "gh auth login."

Using variables

In all of the following examples, you can use variables to simplify your scripts. Use `-F` to pass a variable that is a number, Boolean, or null. Use `-f` for other variables. For example,

```
my_org="octo-org"
my_num=5
```

```
gh api graphql -f query='
  query($organization: String! $number: Int!){
    organization(login: $organization){
      projectV2(number: $number) {
        id
      }
    }
  }' -f organization=$my_org -F number=$my_num
```

For more information, see "Forming calls with GraphQL."

Finding information about projects

Use queries to get data about projects. For more information, see "Forming calls with GraphQL."

Finding the node ID of an organization project

To update your project through the API, you will need to know the node ID of the project.

You can find the node ID of an organization project if you know the organization name and project number. Replace `ORGANIZATION` with the name of your organization. For example, `octo-org`. Replace `NUMBER` with the project number. To find the project number, look at the project URL. For example, `https://github.com/orgs/octo-org/projects/5` has a project number of 5.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"query{organization(login: \"ORGANIZATION\")
{projectV2(number: NUMBER){id}}}"}'
```

```
gh api graphql -f query='
  query{
    organization(login: "ORGANIZATION"){
      projectV2(number: NUMBER) {
        id
      }
    }
  }'
```

You can also find the node ID of all projects in your organization. The following example will return the node ID and title of the first 20 projects in an organization. Replace `ORGANIZATION` with the name of your organization. For example, `octo-org`.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"{organization(login: \"ORGANIZATION\") {projectsV2(first: 20)
{nodes {id title}}}}"}'
```

```
gh api graphql -f query='
  query{
    organization(login: "ORGANIZATION") {
      projectsV2(first: 20) {
        nodes {
          id
          title
        }
      }
    }
```

```
  }'
```

Finding the node ID of a user project

To update your project through the API, you will need to know the node ID of the project.

You can find the node ID of a user project if you know the project number. Replace `USER` with your user name. For example, `octocat`. Replace `NUMBER` with your project number. To find the project number, look at the project URL. For example, `https://github.com/users/octocat/projects/5` has a project number of 5.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"query{user(login: \"USER\") {projectV2(number: NUMBER)
{id}}}"}'
```

```
gh api graphql -f query='
  query{
    user(login: "USER"){
      projectV2(number: NUMBER) {
        id
      }
    }
  }'
```

You can also find the node ID for all of your projects. The following example will return the node ID and title of your first 20 projects. Replace `USER` with your username. For example, `octocat`.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"{user(login: \"USER\") {projectsV2(first: 20) {nodes {id
title}}}}"}'
```

```
gh api graphql -f query='
  query{
    user(login: "USER") {
      projectsV2(first: 20) {
        nodes {
          id
          title
        }
      }
    }
  }'
```

Finding the node ID of a field

To update the value of a field, you will need to know the node ID of the field. Additionally, you will need to know the ID of the options for single select fields and the ID of the iterations for iteration fields.

The following example will return the ID, name, settings, and configuration for the first 20 fields in a project. Replace `PROJECT_ID` with the node ID of your project.

```
curl --request POST \
  --url https://api.github.com/graphql \
```

```
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"query{ node(id: \"PROJECT_ID\") { ... on ProjectV2 {
fields(first: 20) { nodes { ... on ProjectV2Field { id name } ... on
ProjectV2IterationField { id name configuration { iterations { startDate id }}}
... on ProjectV2SingleSelectField { id name options { id name }}}}}}"}'
```

```
gh api graphql -f query='
  query{
  node(id: "PROJECT_ID") {
    ... on ProjectV2 {
      fields(first: 20) {
        nodes {
          ... on ProjectV2Field {
            id
            name
          }
          ... on ProjectV2IterationField {
            id
            name
            configuration {
              iterations {
                startDate
                id
              }
            }
          }
          ... on ProjectV2SingleSelectField {
            id
            name
            options {
              id
              name
            }
          }
        }
      }
    }
  }
}'
```

The response will look similar to the following example:

```
{
  "data": {
    "node": {
      "fields": {
        "nodes": [
          {
            "id": "PVTF_lADOANN5s84ACbL0zgBZrZY",
            "name": "Title"
          },
          {
            "id": "PVTF_lADOANN5s84ACbL0zgBZrZc",
            "name": "Assignees"
          },
          {
            "id": "PVTSSF_lADOANN5s84ACbL0zgBZrZg",
            "name": "Status",
            "options": [
              {
                "id": "f75ad846",
                "name": "Todo"
              },
              {
                "id": "47fc9ee4",
                "name": "In Progress"
              },
              {
```

```
                "id": "98236657",
                "name": "Done"
              }
            ]
          },
          {
            "id": "PVTIF_lADOANN5s84ACbL0zgBah28",
            "name": "Iteration",
            "configuration": {
              "iterations": [
                {
                  "startDate": "2022-05-29",
                  "id": "cfc16e4d"
                }
              ]
            }
          }
        ]
      }
    }
  }
}
```


Each field has an ID and name. Single select fields are returned as a `ProjectV2SingleSelectField` object and have an `options` field where you can find the ID of each option for the single select. Iteration fields are returned as a `ProjectV2IterationField` object and have a `configuration` field which includes an `iterations` field containing the ID and information about each iteration.

If you just need the name and ID of a field, and do not need information about iterations or a single select field's options, you can make use of the `ProjectV2FieldCommon` object.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"query{ node(id: \"PROJECT_ID\") { ... on ProjectV2 {
fields(first: 20) { nodes { ... on ProjectV2FieldCommon { id name }}}}}"}'
```

```
gh api graphql -f query='
  query{
  node(id: "PROJECT_ID") {
    ... on ProjectV2 {
      fields(first: 20) {
        nodes {
          ... on ProjectV2FieldCommon {
            id
            name
          }
        }
      }
    }
  }
}'
```

The response when using the `ProjectV2FieldCommon` object will look similar to the following example:


```
{
  "data": {
    "node": {
      "fields": {
        "nodes": [
          {
            "__typename": "ProjectV2Field",
            "id": "PVTF_lADOANN5s84ACbL0zgBZrZY",
```

```
            "name": "Title"
          },
          {
            "__typename": "ProjectV2Field",
            "id": "PVTF_lADOANN5s84ACbL0zgBZrZc",
            "name": "Assignees"
          },
          {
            "__typename": "ProjectV2SingleSelectField",
            "id": "PVTSSF_lADOANN5s84ACbL0zgBZrZg",
            "name": "Status"
          },
          {
            "__typename": "ProjectV2IterationField",
            "id": "PVTIF_lADOANN5s84ACbL0zgBah28",
            "name": "Iteration"
          }
        ]
      }
    }
  }
}
```


Finding information about items in a project

You can query the API to find information about items in your project.

The following example will return the first 20 issues, pull requests, and draft issues in a project. For issues and pull requests, it will also return title and the first 10 assignees. For draft issue, it will return the title and body. The example will also return the field name and value for any text, date, or single select fields in the first 8 fields of the project. Replace `PROJECT_ID` with the node ID of your project.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"query{ node(id: \"PROJECT_ID\") { ... on ProjectV2 {
items(first: 20) { nodes{ id fieldValues(first: 8) { nodes{ ... on
ProjectV2ItemFieldTextValue { text field { ... on ProjectV2FieldCommon {  name
}}} ... on ProjectV2ItemFieldDateValue { date field { ... on ProjectV2FieldCommon
{ name } } } ... on ProjectV2ItemFieldSingleSelectValue { name field { ... on
ProjectV2FieldCommon { name }}}}} content{ ... on DraftIssue { title body } ...on
Issue { title assignees(first: 10) { nodes{ login }}} ...on PullRequest { title
assignees(first: 10) { nodes{ login }}}}}}}}}"}'
```

```
gh api graphql -f query='
  query{
    node(id: "PROJECT_ID") {
        ... on ProjectV2 {
          items(first: 20) {
            nodes{
              id
              fieldValues(first: 8) {
                nodes{
                  ... on ProjectV2ItemFieldTextValue {
                    text
                    field {
                      ... on ProjectV2FieldCommon {
                        name
                      }
                    }
                  }
                  ... on ProjectV2ItemFieldDateValue {
                    date
                    field {
                      ... on ProjectV2FieldCommon {
```

```
                    name
                  }
                }
              }
              ... on ProjectV2ItemFieldSingleSelectValue {
                name
                field {
                  ... on ProjectV2FieldCommon {
                    name
                  }
                }
              }
            }
          }
          content{
            ... on DraftIssue {
              title
              body
            }
            ...on Issue {
              title
              assignees(first: 10) {
                nodes{
                  login
                }
              }
            }
            ...on PullRequest {
              title
              assignees(first: 10) {
                nodes{
                  login
                }
              }
            }
          }
        }
      }
    }
  }'
```

A project may contain items that a user does not have permission to view. In this case, the item type will be returned as `REDACTED`.

Updating projects

Use mutations to update projects. For more information, see "Forming calls with GraphQL."

> **Note:** You cannot add and update an item in the same call. You must use `addProjectV2ItemById` to add the item and then use `updateProjectV2ItemFieldValue` to update the item.

Adding an item to a project

The following example will add an issue or pull request to your project. Replace `PROJECT_ID` with the node ID of your project. Replace `CONTENT_ID` with the node ID of the issue or pull request that you want to add.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"mutation {addProjectV2ItemById(input: {projectId:
\"PROJECT_ID\" contentId: \"CONTENT_ID\"}) {item {id}}}"}'
```

```
gh api graphql -f query='
  mutation {
    addProjectV2ItemById(input: {projectId: "PROJECT_ID" contentId:
"CONTENT_ID"}) {
      item {
        id
      }
    }
  }'
```

The response will contain the node ID of the newly created item.

```
{
  "data": {
    "addProjectV2ItemById": {
      "item": {
        "id": "PVTI_lADOANN5s84ACbL0zgBVd94"
      }
    }
  }
}
```

If you try to add an item that already exists, the existing item ID is returned instead.

Adding a draft issue to a project

The following example will add a draft issue to your project. Replace `PROJECT_ID` with the node ID of your project. Replace `TITLE` and `BODY` with the content you want for the new draft issue.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"mutation {addProjectV2DraftIssue(input: {projectId:
\"PROJECT_ID\" title: \"TITLE\" body: \"BODY\"}) {projectItem {id}}}"}'
```

```
gh api graphql -f query='
  mutation {
    addProjectV2DraftIssue(input: {projectId: "PROJECT_ID" title: "TITLE" body:
"BODY"}) {
      projectItem {
        id
      }
    }
  }'
```

The response will contain the node ID of the newly created draft issue.

```
{
  "data": {
    "addProjectV2DraftIssue": {
      "projectItem": {
        "id": "PVTI_lADOANN5s84ACbL0zgBbxFc"
      }
    }
  }
}
```

Updating a project's settings

The following example will update your project's settings. Replace `PROJECT_ID` with the node ID of your project. Set `public` to `true` to make your project public on GitHub Enterprise Server. Modify `readme` to make changes to your project's README.

```
curl --request POST \
--url https://api.github.com/graphql \
--header 'Authorization: Bearer TOKEN' \
--data '{"query":"mutation { updateProjectV2(input: { projectId: \"PROJECT_ID\",
title: \"Project title\", public: false, readme: \"# Project README\n\nA long
description\", shortDescription: \"A short description\"}) { projectV2 { id,
title, readme, shortDescription }}}"}'
```

```
gh api graphql -f query='
  mutation {
    updateProjectV2(
      input: {
        projectId: "PROJECT_ID",
        title: "Project title",
        public: false,
        readme: "# Project README\n\nA long description",
        shortDescription: "A short description"
      }
    ) {
      projectV2 {
        id
        title
        readme
        shortDescription
      }
    }
  }'
```

Updating a custom text, number, or date field

The following example will update the value of a text field for an item. Replace `PROJECT_ID` with the node ID of your project. Replace `ITEM_ID` with the node ID of the item you want to update. Replace `FIELD_ID` with the ID of the field that you want to update.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"mutation {updateProjectV2ItemFieldValue( input: { projectId:
\"PROJECT_ID\" itemId: \"ITEM_ID\" fieldId: \"FIELD_ID\" value: { text: \"Updated
text\" }}) { projectV2Item { id }}}"}'
```

```
gh api graphql -f query='
  mutation {
    updateProjectV2ItemFieldValue(
      input: {
        projectId: "PROJECT_ID"
        itemId: "ITEM_ID"
        fieldId: "FIELD_ID"
        value: {
          text: "Updated text"
        }
      }
    ) {
      projectV2Item {
        id
      }
    }
  }'
```

Note: You cannot use `updateProjectV2ItemFieldValue` to change `Assignees`, `Labels`, `Milestone`, or `Repository` because these fields are properties of pull requests and issues, not of project items. Instead, you may use the following mutations:

- addAssigneesToAssignable
- removeAssigneesFromAssignable
- addLabelsToLabelable
- removeLabelsFromLabelable
- updateIssue
- updatePullRequest
- transferIssue

Updating a single select field

The following example will update the value of a single select field for an item.

- `PROJECT_ID` - Replace this with the node ID of your project.
- `ITEM_ID` - Replace this with the node ID of the item you want to update.
- `FIELD_ID` - Replace this with the ID of the single select field that you want to update.
- `OPTION_ID` - Replace this with the ID of the desired single select option.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"mutation {updateProjectV2ItemFieldValue( input: { projectId:
\"PROJECT_ID\" itemId: \"ITEM_ID\" fieldId: \"FIELD_ID\" value: {
singleSelectOptionId: \"OPTION_ID\" }}) { projectV2Item { id }}}"}'
```

```
gh api graphql -f query='
  mutation {
    updateProjectV2ItemFieldValue(
      input: {
        projectId: "PROJECT_ID"
        itemId: "ITEM_ID"
        fieldId: "FIELD_ID"
        value: {
          singleSelectOptionId: "OPTION_ID"
        }
      }
    ) {
      projectV2Item {
        id
      }
    }
  }'
```

Updating an iteration field

The following example will update the value of an iteration field for an item.

- `PROJECT_ID` - Replace this with the node ID of your project.
- `ITEM_ID` - Replace this with the node ID of the item you want to update.
- `FIELD_ID` - Replace this with the ID of the iteration field that you want to update.
- `ITERATION_ID` - Replace this with the ID of the desired iteration. This can be either an active or completed iteration.

```
curl --request POST \
  --url https://api.github.com/graphql \
```

```
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"mutation {updateProjectV2ItemFieldValue( input: { projectId:
\"PROJECT_ID\" itemId: \"ITEM_ID\" fieldId: \"FIELD_ID\" value: { iterationId:
\"ITERATION_ID\" }}) { projectV2Item { id }}}"}'
```

```
gh api graphql -f query='
  mutation {
    updateProjectV2ItemFieldValue(
      input: {
        projectId: "PROJECT_ID"
        itemId: "ITEM_ID"
        fieldId: "FIELD_ID"
        value: {
          iterationId: "ITERATION_ID"
        }
      }
    ) {
      projectV2Item {
        id
      }
    }
  }'
```

Deleting an item from a project

The following example will delete an item from a project. Replace `PROJECT_ID` with the node ID of your project. Replace `ITEM_ID` with the node ID of the item you want to delete.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: Bearer TOKEN' \
  --data '{"query":"mutation {deleteProjectV2Item(input: {projectId:
\"PROJECT_ID\" itemId: \"ITEM_ID\"}) {deletedItemId}}"}'
```

```
gh api graphql -f query='
  mutation {
    deleteProjectV2Item(
      input: {
        projectId: "PROJECT_ID"
        itemId: "ITEM_ID"
      }
    ) {
      deletedItemId
    }
  }'
```

Managing projects

Creating projects

You can use a mutation to create a new project. For more information, see "Forming calls with GraphQL."

To create a new project using the API, you'll need to provide a name for the project and the node ID of a GitHub Enterprise Server user or organization who will become the project's owner.

You can find the node ID of a GitHub Enterprise Server user or organization if you know the username. Replace `GITHUB_OWNER` with the GitHub Enterprise Server username of the

new project owner.

```
curl --request GET \
  --url https://api.github.com/users/GITHUB_OWNER \
  --header 'Authorization: token TOKEN' \
  --header 'Accept: application/vnd.github+json'
```

```
gh api -H "Accept: application/vnd.github+json" /users/GITHUB_OWNER
```

To create the project, replace `OWNER_ID` with the node ID of the new project owner and replace `PROJECT_NAME` with a name for the project.

```
curl --request POST \
  --url https://api.github.com/graphql \
  --header 'Authorization: token TOKEN' \
  --data '{"query":"mutation {createProjectV2(input: {ownerId: \"OWNER_ID\"
title: \"PROJECT_NAME\"}) {projectV2 {id}}}"}'
```

```
gh api graphql -f query='
  mutation{
    createProjectV2(
      input: {
        ownerId: "OWNER_ID",
        title: "PROJECT_NAME"
      }
    ){
      projectV2 {
        id
      }
     }
  }'
```

Using webhooks

You can use webhooks to subscribe to events taking place in your project. For example, when an item is edited, GitHub Enterprise Server can send a HTTP POST payload to the webhook's configured URL which can trigger automation on your server. For more information about webhooks, see "About webhooks." To learn more about the `projects_v2_item` webhook event, see "Webhook events and payloads."