Discovering resources for a user

In this article

Learn how to find the repositories and organizations that your app can access for a user in a reliable way for your authenticated requests to the REST API.

When making authenticated requests to the GitHub API, applications often need to fetch the current user's repositories and organizations. In this guide, we'll explain how to reliably discover those resources.

To interact with the GitHub API, we'll be using Octokit.rb. You can find the complete source code for this project in the platform-samples repository.

Getting started

If you haven't already, you should read the "Basics of Authentication" guide before working through the examples below. The examples below assume that you have registered an OAuth app and that your application has an OAuth token for a user.

Discover the repositories that your app can access for a user

In addition to having their own personal repositories, a user may be a collaborator on repositories owned by other users and organizations. Collectively, these are the repositories where the user has privileged access: either it's a private repository where the user has read or write access, or it's a public or internal repository where the user has write access.

OAuth scopes and organization application policies determine which of those repositories your app can access for a user. Use the workflow below to discover those repositories.

As always, first we'll require GitHub's Octokit.rb Ruby library. Then we'll configure Octokit.rb to automatically handle pagination for us. For more information about pagination, see "Using pagination in the REST API."

```
require 'octokit'

Octokit.auto_paginate = true
```

Next, we'll pass in our application's OAuth token for a given user:

```
# !!! DO NOT EVER USE HARD-CODED VALUES IN A REAL APP !!!
# Instead, set and test environment variables, like below.
client = Octokit::Client.new :access_token => ENV["OAUTH_ACCESS_TOKEN"]
```

Then, we're ready to fetch the repositories that our application can access for the user:

```
client.repositories.each do |repository|
  full_name = repository[:full_name]
  has_push_access = repository[:permissions][:push]

  access_type = if has_push_access
                  "write"
                else
                  "read-only"
                end

  puts "User has #{access_type} access to #{full_name}."
end
```

Discover the organizations that your app can access for a user

Applications can perform all sorts of organization-related tasks for a user. To perform these tasks, the app needs an OAuth authorization with sufficient permission. For example, the `read:org` scope allows you to list teams, and the `user` scope lets you publicize the user’s organization membership. Once a user has granted one or more of these scopes to your app, you're ready to fetch the user’s organizations.

Just as we did when discovering repositories above, we'll start by requiring GitHub's Octokit.rb Ruby library and configuring it to take care of pagination for us. For more information about pagination, see "Using pagination in the REST API."

```
require 'octokit'

Octokit.auto_paginate = true
```

Next, we'll pass in our application's OAuth token for a given user to initialize our API client:

```
# !!! DO NOT EVER USE HARD-CODED VALUES IN A REAL APP !!!
# Instead, set and test environment variables, like below.
client = Octokit::Client.new :access_token => ENV["OAUTH_ACCESS_TOKEN"]
```

Then, we can list the organizations that our application can access for the user:

```
client.organizations.each do |organization|
  puts "User belongs to the #{organization[:login]} organization."
end
```

Return all of the user's organization memberships

If you've read the docs from cover to cover, you may have noticed an API method for listing a user's public organization memberships. Most applications should avoid this API method. This method only returns the user's public organization memberships, not their private organization memberships.

As an application, you typically want all of the user's organizations that your app is authorized to access. The workflow above will give you exactly that.