Migrating OAuth apps to GitHub Apps

In this article

Learn about the advantages of migrating your OAuth app to a GitHub App, and learn how to migrate your OAuth app.

Benefits of migrating from OAuth apps to GitHub Apps

GitHub Apps are the recommended way to integrate with GitHub. GitHub Apps offer many advantages over OAuth apps, including:

- enhanced security features, like fine-grained permissions, choice over repository access, and short lived tokens
- the ability to act independently of or on behalf of a user
- scalable rate limits
- built-in webhooks

For more information, see "About creating GitHub Apps."

Converting an OAuth app to a GitHub App

The following steps provide an overview of how to migrate from an OAuth app to a GitHub App. The specific steps depend on your app.

1. Review your OAuth app

Re-familiarize yourself with the code for your OAuth app. The API requests that your OAuth app makes will help you decide what permissions to select for your GitHub App.

Additionally, there are a few REST API endpoints that are not available for OAuth apps. Verify that any REST endpoints that you use are available for GitHub Apps by reviewing "Endpoints available for GitHub App installation access tokens."

2. Register a GitHub App

Register a new GitHub App. For more information, see "Registering a GitHub App."

Compared to an OAuth app, you have more control over GitHub App settings. Some key additions are:

- Unlike an OAuth app, which always acts on behalf of a user, you can make your GitHub App take actions as itself or on behalf of a user. If you do not want your new GitHub App to take actions on behalf of a user, you can skip the "Identifying and authorizing users" settings. For more information, see "About authentication with a GitHub App."

- You can use webhooks to notify your GitHub App when specific events occur. Unlike webhooks for OAuth apps, which you must configure via the API for each repository or organization, webhooks are built into GitHub Apps. When you register your GitHub App, you can select the webhook events that you want to receive. Additionally, if your OAuth app currently uses polling to determine if an event had occurred, consider subscribing to webhooks instead to help your GitHub App stay within the rate limit. For more information, see "Using webhooks with GitHub Apps."
- With an OAuth app, you request scopes when a user authorizes your app. With a GitHub App, you specify permissions in the app settings. These permissions are more granular than scopes and enable you to only select the permissions that your app needs. Additionally, these permissions are mapped to REST API endpoints and webhook events, so you can easily determine what permissions your GitHub App needs in order to access a specific REST API endpoint or subscribe to a specific webhook. Permissions are not currently documented for GraphQL requests. For more information, see "Choosing permissions for a GitHub App."

3. Modify the code for your app

Once you have registered a GitHub App, adapt the code from your old OAuth app to work with your new GitHub App.

Update authentication

You will need to update your app's code to handle API authentication for your GitHub App. A GitHub App can authenticate in three ways:

- As the app itself, in order to get or modify details about the GitHub App registration or to create an installation access token. For more information, see "Authenticating as a GitHub App."
- As an app installation, in order to take actions on behalf of itself. For more information, see "Authenticating as a GitHub App installation."
- On behalf of a user, in order to attribute actions to a user. For more information, see "Authenticating with a GitHub App on behalf of a user."

If you are using GitHub's official Octokit.js library, you can use the built-in `App` object to authenticate. For examples, see "Scripting with the REST API and JavaScript" and "Building a GitHub App that responds to webhook events."

Review rate limits

Review the differences in rate limits between OAuth apps and GitHub Apps. GitHub Apps use sliding rules for rate limits, which can increase based on the number of repositories and number of users in the organization. For more information, see "Rate limits for GitHub Apps."

If possible, consider using conditional requests and subscribing to webhooks instead of polling to help you stay within rate limits. For more information about conditional requests, see "Resources in the REST API." For more information about using webhooks with your GitHub App, see "Using webhooks with GitHub Apps" and "Building a GitHub App that responds to webhook events."

Test your code

Test your new GitHub App to make sure that your code works as expected.

4. Publicize your new GitHub App

If you want other accounts to be able to use your new GitHub App, make sure that your

app is public. If you want to make your GitHub App more discoverable, list your app in GitHub Marketplace. For more information, see "About GitHub Marketplace for apps" and "Making a GitHub App public or private."

5. Instruct your users to migrate

Once your new GitHub App is ready, instruct users of your old OAuth app to migrate to your new GitHub App. There is not a way to automatically migrate your users. Each user must install and/or authorize your GitHub App on their own.

As the app owner, you should include calls to action to encourage your users to install/authorize the new GitHub App and revoke authorization for the old OAuth app. You should also update any documentation or user interface elements.

Prompt users to install your GitHub App

If you want your GitHub App to make API requests on behalf of itself or access organization or repository resources, the user must install your GitHub App. When a user installs a GitHub App on their account or organization, they choose which repositories the app can access, and they grant the app the organization and repository permissions that it requested.

To help your users install your GitHub App, you can add a link to your app's webpage that users can click to install the GitHub App. The format of the install URL is `https://github.com/apps/YOUR_APP_NAME/installations/new`. Replace `YOUR_APP_NAME` with the sluggified name of your GitHub App, which you can find in the "Public link" field on the settings page for your GitHub App.

To pre-select any repositories your OAuth app had access to, you can append `/permissions` and query parameters to the install URL. This helps users grant your GitHub App access to repositories that your OAuth app already has access to. The query parameters are:

- `suggested_target_id`: The ID of the user or organization that is installing your GitHub App. This parameter is required.
- `repository_ids[]`: The repository IDs to select for the installation. If omitted, all repositories are selected. The maximum number of repositories that can be pre-selected is 100. To get a list of repositories that your OAuth app has access to, use the List repositories for the authenticated user and List organization repositories endpoints.

For example: `https://github.com/apps/YOUR_APP_NAME/installations/new/permissions?suggested_target_id=ID_OF_USER_OR_ORG&repository_ids[]=REPO_A_ID&repository_ids[]=REPO_B_ID`.

For more information about installing GitHub Apps, see "Installing a GitHub App from GitHub Marketplace for your personal account," "Installing a GitHub App from GitHub Marketplace for your organizations," "Installing a GitHub App from a third party" and "Installing your own GitHub App."

Prompt users to authorize your app

If you want your GitHub App to make API requests on behalf of a user, the user must authorize the app. When a user authorizes an app, they grant the app permission to act on their behalf, and they grant the account permissions that the app requested. If the app is installed on an organization account, each user within that organization must authorize the app in order for the app to act on their behalf.

To prompt users to authorize your app, you will lead them through the web application flow or device flow. For more information, see "Generating a user access token for a

GitHub App."

For more information about authorizing GitHub Apps, see "Authorizing GitHub Apps."

Encourage your users to revoke OAuth app access

You should also encourage your users to revoke access for your old OAuth app. This will help you fully transition away from your OAuth app and will help keep your users' data secure. For more information, see "Reviewing your authorized OAuth apps."

Update any interfaces or documentation

You should update any user interface or documentation related to your app to reflect the change from an OAuth app to GitHub App.

6. Remove webhooks for your old OAuth app

When a user installs your GitHub App and grants access to a repository, you should remove any webhooks for your old OAuth app. If your new GitHub App and your old OAuth app respond to webhooks for the same event, the user may observe duplicate behavior.

To remove repository webhooks, you can listen for the `installation_repositories` webhook with the `added` action. When your GitHub App receives that event, you can use the REST API to delete the webhook on those repositories for your OAuth app. For more information, see "Webhook events and payloads" and "Repository webhooks."

Similarly, to remove organization webhooks, you can listen for the `installation` webhook with the `created` action. When your GitHub App receives that event for an organization, you can use the REST API to delete the webhook on that organization and corresponding repositories for your OAuth app. For more information, see "Webhook events and payloads," "Organization webhooks," and "Repository webhooks."

7. Delete your old OAuth app

Once your users have migrated to your new GitHub App, you should delete your old OAuth app. This will help avoid abuse of the OAuth app's credentials. This action will also revoke all of the OAuth app's remaining authorizations. For more information, see "Deleting an OAuth app." If your OAuth app is listed on GitHub Marketplace, you may need to contact GitHub Support to remove your app from the marketplace first.