Authenticating as a GitHub App installation

In this article

You can make your GitHub App authenticate as an installation in order to make API requests that affect resources owned by the account where the app is installed.

About authentication as a GitHub App installation

Once your GitHub App is installed on an account, you can make it authenticate as an app installation for API requests. This allows the app to access resources owned by that installation, as long as the app was granted the necessary repository access and permissions. API requests made by an app installation are attributed to the app. For more information about installing GitHub Apps, see "Installing GitHub Apps."

For example, if you want your app to change the `Status` field of an issue on a project owned by an organization called "octo-org," then you would authenticate as the octo-org installation of your app. The timeline of the issue would state that your app updated the status.

To make an API request as an installation, you must first generate an installation access token. Then, you will send the installation access token in the `Authorization` header of your subsequent API requests. You can also use GitHub's Octokit SDKs, which can generate an installation access token for you.

If a REST API endpoint works with a GitHub App installation access token, the REST reference documentation for that endpoint will say "Works with GitHub Apps." Additionally, your app must have the required permissions to use the endpoint. For more information, see "Choosing permissions for a GitHub App."

App installations can also use the GraphQL API. Similar to the REST API, the app must have certain permissions to access objects in the GraphQL API. For GraphQL requests, you should test you app to ensure that your app has the required permissions for the GraphQL queries and mutations that you want to make.

You can also use an installation access token to authenticate for HTTP-based Git access. You app must have the "Contents" repository permission. You can then use the installation access token as the HTTP password. Replace `TOKEN` with the installation access token: `git clone https://x-access-token:TOKEN@github.com/owner/repo.git`.

Requests made with an installation access token are sometimes called "server-to-server" requests.

For more information about authenticating as an app on behalf of a user instead of as an app installation, see "Authenticating with a GitHub App on behalf of a user".

Using an installation access token to authenticate as an app installation

To authenticate as an installation with an installation access token, first use the REST API to generate an installation access token. Then, use that installation access token in the `Authorization` header of a REST API or GraphQL API request. The installation access token will expire after 1 hour.

Generating an installation access token

1. Generate a JSON web token (JWT) for your app. For more information, see "Generating a JSON Web Token (JWT) for a GitHub App".
2. Get the ID of the installation that you want to authenticate as.

   If you are responding to a webhook event, the webhook payload will include the installation ID.

   You can also use the REST API to find the ID for an installation of your app. For example, you can get an installation ID with the `GET /users/{username}/installation`, `GET /repos/{owner}/{repo}/installation`, `GET /orgs/{org}/installation`, or `GET /app/installations` endpoints. For more information, see "GitHub Apps".
3. Send a REST API `POST` request to `/app/installations/INSTALLATION_ID/access_tokens`. Include your JSON web token in the `Authorization` header of your request. Replace `INSTALLATION_ID` with the ID of the installation that you want to authenticate as.

   For example, send this curl request. Replace `INSTALLATION_ID` with the ID of the installation and `JWT` with your JSON web token:

   ```
   curl --request POST \
   --url "http(s)://HOSTNAME/api/v3/app/installations/INSTALLATION_ID/access_toker
   --header "Accept: application/vnd.github+json" \
   --header "Authorization: Bearer JWT"
   ```

   Optionally, you can use the `repositories` or `repository_ids` body parameters to specify individual repositories that the installation access token can access. If you don't use `repositories` or `repository_ids` to grant access to specific repositories, the installation access token will have access to all repositories that the installation was granted access to. The installation access token cannot be granted access to repositories that the installation was not granted access to.

   Optionally, use the `permissions` body parameter to specify the permissions that the installation access token should have. If `permissions` is not specified, the installation access token will have all of the permissions that were granted to the app. The installation access token cannot be granted permissions that the app was not granted.

   The response will include an installation access token, the time that the token expires, the permissions that the token has, and the repositories that the token can access. The installation access token will expire after 1 hour.

For more information about this endpoint, see "GitHub Apps".

> **Note:** In most cases, you can use `Authorization: Bearer` or `Authorization: token` to pass a token. However, if you are passing a JSON web token (JWT), you must use `Authorization: Bearer`.

Authenticating with an installation access token

To authenticate with an installation access token, include it in the `Authorization` header of an API request. The access token will work with both the GraphQL API and the REST API.

Your app must have the required permissions to use the endpoint. For more information, see "Choosing permissions for a GitHub App."

In the following example, replace `INSTALLATION_ACCESS_TOKEN` with an installation access token:

```
curl --request GET \
--url "http(s)://HOSTNAME/api/v3/meta" \
--header "Accept: application/vnd.github+json" \
--header "Authorization: Bearer INSTALLATION_ACCESS_TOKEN"
```

Using the Octokit.js SDK to authenticate as an app installation

You can use GitHub's Octokit.js SDK to authenticate as an app installation. One advantage of using the SDK to authenticate is that you do not need to generate a JSON web token (JWT) yourself. Additionally, the SDK will take care of regenerating an installation access token for you so you don't need to worry about the one hour expiration.

> You must install and import `octokit` in order to use the Octokit.js library. The following example uses import statements in accordance with ES6. For more information about different installation and import methods, see the Octokit.js README's Usage section.

Using Octokit.js to authenticate with an installation ID

1. Get the ID of your GitHub App. You can find your app's ID on the settings page for your GitHub App. For more information about navigating to the settings page for your GitHub App, see "Modifying a GitHub App registration."
2. Generate a private key. For more information, see "Managing private keys for GitHub Apps".
3. Get the ID of the installation that you want to authenticate as.

   If you are responding to a webhook event, the webhook payload will include the installation ID.

   You can also use the REST API to find the ID for an installation of your app. For example, you can get an installation ID with the `GET /users/{username}/installation`, `GET /repos/{owner}/{repo}/installation`, `GET /orgs/{org}/installation`, or `GET /app/installations` endpoints. For more information, see "GitHub Apps".

4. Import `App` from `octokit`. Create a new instance of `App`. In the following example, replace `APP_ID` with a reference to your app's ID. Replace `PRIVATE_KEY` with a reference to your app's private key.

```javascript
import { App } from "octokit";

const app = new App({
  appId: APP_ID,
  privateKey: PRIVATE_KEY,
});
```

5. Use the `getInstallationOctokit` method to create an authenticated `octokit` instance. In the following example, replace `INSTALLATION_ID` with the ID of the installation of your app that you want to authenticate on behalf of.

```javascript
const octokit = await app.getInstallationOctokit(INSTALLATION_ID);
```

6. Use an `octokit` method to make a request to the API.

   Your app must have the required permissions to use the endpoint. For more information, see "Choosing permissions for a GitHub App."

   For example, to make a request to the GraphQL API:

```javascript
await octokit.graphql(`
  query {
    viewer {
      login
    }
  }
  `)
```

   For example, to make a request to the REST API:

```javascript
await octokit.request("GET /meta")
```

Using Octokit.js to authenticate in response to a webhook event

The Octokit.js SDK also passes a pre-authenticated `octokit` instance to webhook event handlers.

1. Get the ID of your GitHub App. You can find your app's ID on the settings page for your GitHub App. For more information about navigating to the settings page for your GitHub App, see "Modifying a GitHub App registration."

2. Generate a private key. For more information, see "Managing private keys for GitHub Apps".

3. Get the webhook secret that you specified in your app's settings. For more information about webhook secrets, see "Using webhooks with GitHub Apps."

4. Import `App` from `octokit`. Create a new instance of `App`. In the following example, replace `APP_ID` with a reference to your app's ID. Replace `PRIVATE_KEY` with a reference to your app's private key. Replace `WEBHOOK_SECRET` with the your app's webhook secret.

JavaScript

```
import { App } from "octokit";

const app = new App({
  appId: APP_ID,
  privateKey: PRIVATE_KEY,
  webhooks: { WEBHOOK_SECRET },
});
```

5. Use an `app.webhooks.*` method to handle webhook events. For more information, see the Octokit.js README's Webhooks section. For example, to create a comment on an issue when the issue is opened:

```
app.webhooks.on("issues.opened", ({ octokit, payload }) => {
  await octokit.request("POST /repos/{owner}/{repo}/issues/{issue_number}/comme
      owner: payload.repository.owner.login,
      repo: payload.repository.name,
      issue_number: payload.issue.number,
      body: `This is a bot post in response to this issue being opened.`,

    }
  )
});
```