Differences between GitHub Apps and OAuth apps

In this article

In general, GitHub Apps are preferred to OAuth apps because they use fine-grained permissions, give more control over which repositories the app can access, and use short-lived tokens.

About GitHub Apps and OAuth apps

In general, GitHub Apps are preferred over OAuth apps. GitHub Apps use fine grained permissions, give the user more control over which repositories the app can access, and use short-lived tokens. These properties can harden the security of your app by limiting the damage that could be done if your app's credentials were leaked.

Similar to OAuth apps, GitHub Apps can still use OAuth 2.0 and generate a type of OAuth token (called a user access token) and take actions on behalf of a user. However, GitHub Apps can also act independently of a user. This is beneficial for automations that do not require user input. The app will continue to work even if the person who installed the app on an organization leaves the organization.

GitHub Apps have built-in, centralized webhooks. GitHub Apps can receive webhook events for all repositories and organizations the app can access. Conversely, OAuth apps must configure webhooks individually for each repository and organization.

The rate limit for GitHub Apps using an installation access token scales with the number of repositories and number of organization users. Conversely, OAuth apps have lower rate limits and do not scale.

There is one case where an OAuth app is preferred over a GitHub App. If your app needs to access enterprise-level resources such as the enterprise object itself, you should use an OAuth app because a GitHub App cannot yet be given permissions against an enterprise. GitHub Apps can still access enterprise-owned organization and repository resources.

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

For more information about migrating an existing OAuth app to a GitHub App, see "Migrating OAuth apps to GitHub Apps."

Who can install GitHub Apps and authorize OAuth apps?

You can install GitHub Apps in your personal account or organizations you own. If you have admin permissions in a repository, you can install GitHub Apps on organization accounts. If a GitHub App is installed in a repository and requires organization permissions, the organization owner must approve the application.

By default, only organization owners can manage the settings of GitHub Apps in an organization. To allow additional users to change the developer settings of GitHub Apps owned by the organization, an owner can grant them GitHub App manager permissions. GitHub App Managers can't manage third-party applications. For more information about adding and removing GitHub App managers in your organization, see "Roles in an organization."

By contrast, users authorize OAuth apps, which gives the app the ability to act as the authenticated user. For example, you can authorize an OAuth app that finds all notifications for the authenticated user. You can always revoke permissions from an OAuth app.

Organization owners can choose whether to allow outside collaborators to request access for unapproved OAuth apps and GitHub Apps. For more information, see "Limiting OAuth app and GitHub App access requests."

Warning: Revoking all permission from an OAuth app deletes any SSH keys the application generated on behalf of the user, including deploy keys.

GitHub Apps	OAuth apps
You must be an organization owner or have admin permissions in a repository to install a GitHub App on an organization. If a GitHub App is installed in a repository and requires organization permissions, the organization owner must approve the application.	You can authorize an OAuth app to have access to resources.
You can install a GitHub App on your personal repository.	You can authorize an OAuth app to have access to resources.
You must be an organization owner, personal repository owner, or have admin permissions in a repository to uninstall a GitHub App and remove its access.	You can delete an OAuth access token to remove access.
You must be an organization owner or have admin permissions in a repository to request a GitHub App installation.	If an organization application policy is active, any organization member can request to install an OAuth app on an organization. An organization owner must approve or deny the request.

What can GitHub Apps and OAuth apps access?

Account owners can use a GitHub App in one account without granting access to another. For example, you can install a third-party build service on your employer's organization, but decide not to grant that build service access to repositories in your personal account. A GitHub App remains installed if the person who set it up leaves the organization.

An *authorized* OAuth app has access to all of the user's or organization owner's

accessible resources.

GitHub Apps	OAuth apps
Installing a GitHub App grants the app access to a user or organization account's chosen repositories.	Authorizing an OAuth app grants the app access to the user's accessible resources. For example, repositories they can access.
The installation token from a GitHub App loses access to resources if an admin removes repositories from the installation.	An OAuth access token loses access to resources when the user loses access, such as when they lose write access to a repository.
Installation access tokens are limited to specified repositories with the permissions chosen by the creator of the app.	An OAuth access token is limited via scopes.
GitHub Apps can request separate access to issues and pull requests without accessing the actual contents of the repository.	OAuth apps need to request the `repo` scope to get access to issues, pull requests, or anything owned by the repository.
GitHub Apps aren't subject to organization application policies. A GitHub App only has access to the repositories an organization owner has granted.	If an organization application policy is active, only an organization owner can authorize the installation of an OAuth app. If installed, the OAuth app gains access to anything visible to the token the organization owner has within the approved organization.
A GitHub App receives a webhook event when an installation is changed or removed. This tells the app creator when they've received more or less access to an organization's resources.	OAuth apps can lose access to an organization or repository at any time based on the granting user's changing access. The OAuth app will not inform you when it loses access to a resource.

Token-based identification

> **Note:** GitHub Apps can also use a user-based token. For more information, see "Authenticating with a GitHub App on behalf of a user."

GitHub Apps	OAuth apps
A GitHub App can request an installation access token by using a private key with a JSON web token format out-of-band.	An OAuth app can exchange a request token for an access token after a redirect via a web request.
An installation token identifies the app as the GitHub Apps bot, such as @jenkins-bot.	An access token identifies the app as the user who granted the token to the app, such as @octocat.
Installation access tokens expire after a predefined amount of time (currently 1 hour).	OAuth tokens remain active until they're revoked by the customer.
GitHub Apps installed on organizations or repositories are subject to rate limits that scale with the number of installations. For more information, see "Rate limits for GitHub Apps."	OAuth tokens use the user's rate limit of 5,000 requests per hour.
Rate limit increases can be granted both at the GitHub Apps level (affecting all installations) and at the individual installation level.	Rate limit increases are granted per OAuth app. Every token granted to that OAuth app gets the increased limit.
GitHub Apps can authenticate on behalf of the user. The flow to authorize is the same as the	The OAuth flow used by OAuth apps authorizes an OAuth app on behalf of the user. This is the

OAuth app authorization flow. User access tokens can expire and be renewed with a refresh token. For more information, see "Refreshing user access tokens" and "Authenticating with a GitHub App on behalf of a user."	an OAuth app on behalf of the user. This is the same flow used to generate a GitHub App user access token.

Requesting permission levels for resources

Unlike OAuth apps, GitHub Apps have targeted permissions that allow them to request access only to what they need. For example, a Continuous Integration (CI) GitHub App can request read access to repository content and write access to the status API. Another GitHub App can have no read or write access to code but still have the ability to manage issues, labels, and milestones. OAuth apps can't use granular permissions.

Access	GitHub Apps (`read` or `write` permissions)	OAuth apps
For access to public repositories	Public repository needs to be chosen during installation.	`public_repo` scope.
For access to repository code/contents	Repository contents	`repo` scope.
For access to issues, labels, and milestones	Issues	`repo` scope.
For access to pull requests, labels, and milestones	Pull requests	`repo` scope.
For access to commit statuses (for CI builds)	Commit statuses	`repo:status` scope.
For access to deployments and deployment statuses	Deployments	`repo_deployment` scope.
To receive events via a webhook	A GitHub App includes a webhook by default.	`write:repo_hook` or `write:org_hook` scope.

Repository discovery

GitHub Apps	OAuth apps
GitHub Apps can look at `/installation/repositories` to see repositories the installation can access.	OAuth apps can look at `/user/repos` for a user view or `/orgs/:org/repos` for an organization view of accessible repositories.
GitHub Apps receive webhooks when repositories are added or removed from the installation.	OAuth apps create organization webhooks for notifications when a new repository is created within an organization.

Webhooks

GitHub Apps	OAuth apps
By default, GitHub Apps have a single webhook that receives the events they are configured to receive for every repository they have access to.	OAuth apps request the webhook scope to create a repository webhook for each repository they need to receive events from.

GitHub Apps receive certain organization-level events with the organization member's permission.	OAuth apps request the organization webhook scope to create an organization webhook for each organization they need to receive organization-level events from.
Webhooks are automatically disabled when the GitHub App is uninstalled.	Webhooks are not automatically disabled if an OAuth app's access token is deleted, and there is no way to clean them up automatically. You will have to ask users to do this manually.

Git access

GitHub Apps	OAuth apps
GitHub Apps ask for repository contents permission and use your installation access token to authenticate via HTTP-based Git. For more information, see "Generating an installation access token for a GitHub App"	OAuth apps ask for `write:public_key` scope and Create a deploy key via the API. You can then use that key to perform Git commands.
The token is used as the HTTP password.	The token is used as the HTTP username.

Machine vs. bot accounts

Machine user accounts are OAuth-based personal accounts that segregate automated systems using GitHub's user system.

Bot accounts are specific to GitHub Apps and are built into every GitHub App.

GitHub Apps	OAuth apps
GitHub App bots do not consume a GitHub Enterprise seat.	A machine user account consumes a GitHub Enterprise seat.
Because a GitHub App bot is never granted a password, a customer can't sign into it directly.	A machine user account is granted a username and password to be managed and secured by the customer.