Managing your personal access tokens

In this article

About personal access tokens
Creating a fine-grained personal access token
Creating a personal access token (classic)
Deleting a personal access token
Using a personal access token on the command line
Further reading

You can use a personal access token in place of a password when authenticating to GitHub in the command line or with the API.

Warning: Treat your access tokens like passwords. For more information, see "Keeping your personal access tokens secure."

About personal access tokens

Personal access tokens are an alternative to using passwords for authentication to GitHub Enterprise Server when using the GitHub API or the command line.

Personal access tokens are intended to access GitHub resources on behalf of yourself. To access resources on behalf of an organization, or for long-lived integrations, you should use a GitHub App. For more information, see "About creating GitHub Apps."

Types of personal access tokens

GitHub currently supports two types of personal access tokens: fine-grained personal access tokens and personal access tokens (classic). GitHub recommends that you use fine-grained personal access tokens instead of personal access tokens (classic) whenever possible.

Both fine-grained personal access tokens and personal access tokens (classic) are tied to the user who generated them and will become inactive if the user loses access to the resource.

Organization owners can set a policy to restrict the access of personal access tokens (classic) to their organization, and enterprise owners can restrict the access of personal access tokens (classic) to the enterprise or organizations owned by the enterprise. For more information, see "Setting a personal access token policy for your organization."

Fine-grained personal access tokens

Fine-grained personal access tokens have several security advantages over personal access tokens (classic):

- Each token can only access resources owned by a single user or organization.

- Each token can only access specific repositories.
- Each token is granted specific permissions, which offer more control than the scopes granted to personal access tokens (classic).
- Each token must have an expiration date.
- Organization owners can require approval for any fine-grained personal access tokens that can access resources in the organization.
- Enterprise owners can require approval for any fine-grained personal access tokens that can access resources in organizations owned by the enterprise.

Personal access tokens (classic)

Personal access tokens (classic) are less secure. However, some features currently will only work with personal access tokens (classic):

- Only personal access tokens (classic) have write access for public repositories that are not owned by you or an organization that you are not a member of.
- Only personal access tokens (classic) automatically have write access for internal repositories that are owned by your enterprise. Fine-grained personal access tokens must be granted access to internal repositories.
- Outside collaborators can only use personal access tokens (classic) to access organization repositories that they are a collaborator on.
- Only personal access tokens (classic) can access enterprises. (Fine-grained personal access token can access organizations owned by enterprises.)
- Some REST API operations are not available to fine-grained personal access tokens. For a list of REST API operations that are supported for fine-grained personal access tokens, see "Endpoints available for fine-grained personal access tokens".

If you choose to use a personal access token (classic), keep in mind that it will grant access to all repositories within the organizations that you have access to, as well as all personal repositories in your personal account.

Keeping your personal access tokens secure

Personal access tokens are like passwords, and they share the same inherent security risks. Before creating a new personal access token, consider if there is a more secure method of authentication available to you:

- To access GitHub from the command line, you can use GitHub CLI or Git Credential Manager instead of creating a personal access token.
- When using a personal access token in a GitHub Actions workflow, consider whether you can use the built-in `GITHUB_TOKEN` instead. For more information, see "Automatic token authentication."

If these options are not possible, and you must create a personal access token, consider using another CLI service to store your token securely.

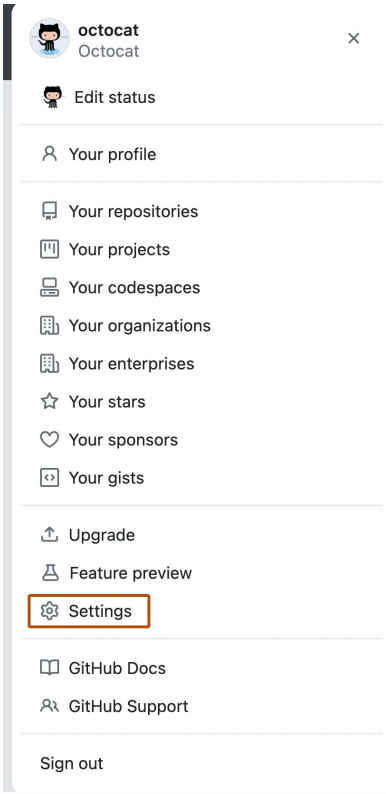
When using a personal access token in a script, you can store your token as a secret and run your script through GitHub Actions. For more information, see "Using secrets in GitHub Actions."

For more information about best practices, see "Keeping your API credentials secure."

Creating a fine-grained personal access token

> **Note**: Fine-grained personal access token are currently in beta and subject to change. To leave feedback, see the feedback discussion.

1. In the upper-right corner of any page, click your profile photo, then click **Settings**.



2. In the left sidebar, click <> **Developer settings**.
3. In the left sidebar, under **Personal access tokens**, click **Fine-grained tokens**.
4. Click **Generate new token**.
5. Under **Token name**, enter a name for the token.
6. Under **Expiration**, select an expiration for the token.
7. Optionally, under **Description**, add a note to describe the purpose of the token.
8. Under **Resource owner**, select a resource owner. The token will only be able to access resources owned by the selected resource owner. Organizations that you are a member of will not appear unless the organization opted in to fine-grained personal access tokens. For more information, see "Setting a personal access token policy for your organization."
9. Optionally, if the resource owner is an organization that requires approval for fine-grained personal access tokens, below the resource owner, in the box, enter a justification for the request.
10. Under **Repository access**, select which repositories you want the token to access. You should choose the minimal repository access that meets your needs. Tokens always include read-only access to all public repositories on GitHub.
11. If you selected **Only select repositories** in the previous step, under the **Selected repositories** dropdown, select the repositories that you want the token to access.
12. Under **Permissions**, select which permissions to grant the token. Depending on which resource owner and which repository access you specified, there are repository, organization, and account permissions. You should choose the minimal permissions necessary for your needs. For more information about what permissions are required for each REST API operation, see "Permissions required for fine-grained personal access tokens."

13. Click **Generate token**.

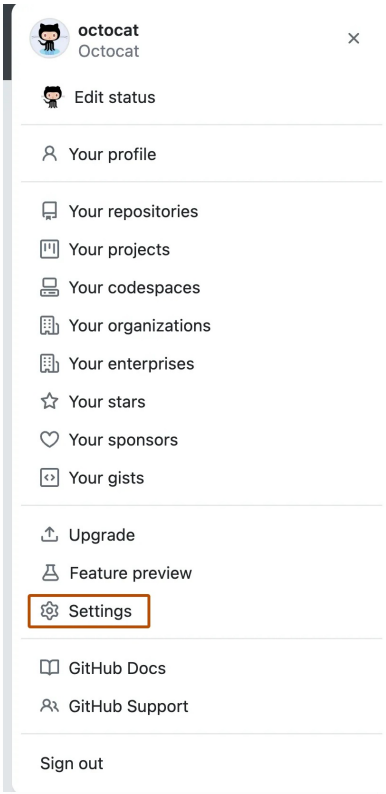
If you selected an organization as the resource owner and the organization requires approval for fine-grained personal access tokens, then your token will be marked as `pending` until it is reviewed by an organization administrator. Your token will only be able to read public resources until it is approved. If you are an owner of the organization, your request is automatically approved. For more information, see "Reviewing and revoking personal access tokens in your organization."

Creating a personal access token (classic)

Note: Organization owners can restrict the access of personal access token (classic) to their organization. If you try to use a personal access token (classic) to access resources in an organization that has disabled personal access token (classic) access, your request will fail with a 403 response. Instead, you must use a GitHub App, OAuth app, or fine-grained personal access token.

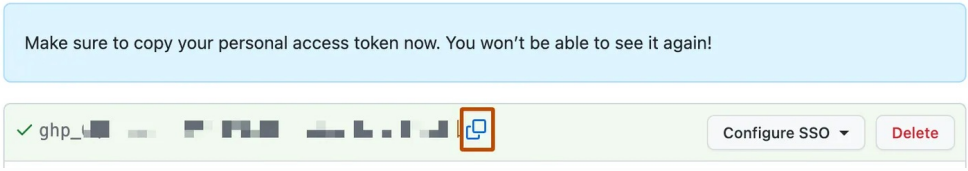
Note: Your personal access token (classic) can access every repository that you can access. GitHub recommends that you use fine-grained personal access tokens instead, which you can restrict to specific repositories. Fine-grained personal access tokens also enable you to specify fine-grained permissions instead of broad scopes.

1. In the upper-right corner of any page, click your profile photo, then click **Settings**.



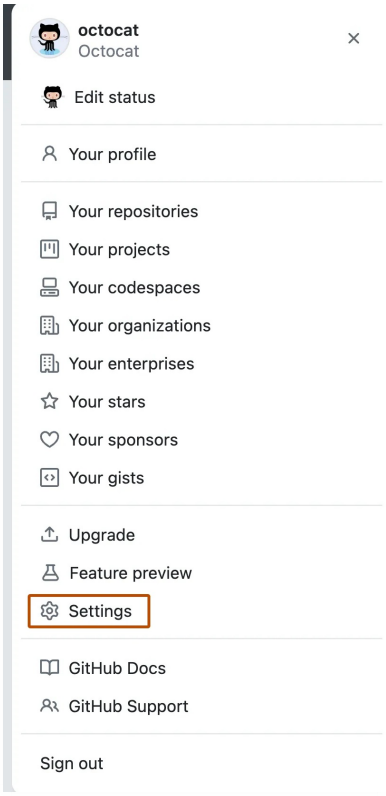
2. In the left sidebar, click **Developer settings**.
3. In the left sidebar, under **Personal access tokens**, click **Tokens (classic)**.
4. Select **Generate new token**, then click **Generate new token (classic)**.
5. In the "Note" field, give your token a descriptive name.

6. To give your token an expiration, select **Expiration**, then choose a default option or click **Custom** to enter a date.
7. Select the scopes you'd like to grant this token. To use your token to access repositories from the command line, select **repo**. A token with no assigned scopes can only access public information. For more information, see "Scopes for OAuth apps."
8. Click **Generate token**.
9. Optionally, to copy the new token to your clipboard, click .



Deleting a personal access token

1. In the upper-right corner of any page, click your profile photo, then click **Settings**.



2. In the left sidebar, click **Developer settings**.
3. In the left sidebar, under **Personal access tokens**, click either **Fine-grained tokens** or **Tokens (classic)**, depending on which type of personal access token you'd like to delete.
4. To the right of the personal access token you want to delete, click **Delete**.

Using a personal access token on the command line

Once you have a personal access token, you can enter it instead of your password when performing Git operations over HTTPS.

For example, to clone a repository on the command line you would enter the following `git clone` command. You would then be prompted to enter your username and password. When prompted for your password, enter your personal access token instead of a password.

```
$ git clone https://HOSTNAME/USERNAME/REPO.git
Username: YOUR_USERNAME
Password: YOUR_PERSONAL_ACCESS_TOKEN
```

Personal access tokens can only be used for HTTPS Git operations. If your repository uses an SSH remote URL, you will need to switch the remote from SSH to HTTPS.

If you are not prompted for your username and password, your credentials may be cached on your computer. You can update your credentials in the Keychain to replace your old password with the token.

Instead of manually entering your personal access token for every HTTPS Git operation, you can cache your personal access token with a Git client. Git will temporarily store your credentials in memory until an expiry interval has passed. You can also store the token in a plain text file that Git can read before every request. For more information, see "Caching your GitHub credentials in Git."

Further reading

- "About authentication to GitHub"
- "Token expiration and revocation"

Legal

© 2023 GitHub, Inc. Terms Privacy Status Pricing Expert services Blog