Managing your personal access tokens

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](#keeping-your-personal-access-tokens-secure).

About personal access tokens

Personal access tokens are an alternative to using passwords for authentication to GitHub when using the [GitHub API](/en/rest/overview/authenticating-to-the-rest-api) or the [command line](#using-a-personal-access-token-on-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](/en/apps/creating-github-apps/setting-up-a-github-app/about-creating-github-apps).

A token has the same capabilities to access resources and perform actions on those resources that the owner of the token has, and is further limited by any scopes or permissions granted to the token. A token cannot grant additional access capabilities to a user. For example, a personal access token can be configured with an `admin:org` scope, but if the owner of the token is not an organization owner, the token will not give administrative access to the organization.

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.

> \[!NOTE]
> Fine-grained personal access tokens, while more secure and controllable, cannot accomplish every task that a personal access token (classic) can. See the section on [Fine-grained personal access tokens limitations](#fine-grained-personal-access-tokens-limitations) below to learn more.

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. For more information, see [Setting a personal access token policy for your organization](/en/organizations/managing-programmatic-access-to-your-organization/setting-a-personal-access-token-policy-for-your-organization#restricting-access-by-personal-access-tokens).

Fine-grained personal access tokens

Fine-grained personal access tokens have several security advantages over personal access tokens (classic), but also have limitations that may prevent you from using them in every scenario. These limits, and our plans to fix them, can be found in the [section below](#fine-grained-personal-access-tokens-limitations).

If you can use a fine-grained personal access token for your scenario, you'll benefit from these improvements:

* Each token is limited to access resources owned by a single user or organization.
* Each token can be further limited to only access specific repositories for that user or organization.
* Each token is granted specific, fine-grained permissions, which offer more control than the scopes granted to personal access tokens (classic).
* Organization owners can require approval for any fine-grained personal access tokens that can access resources in the organization.

Fine-grained personal access tokens limitations

Fine-grained personal access tokens do not support every feature of personal access tokens (classic). These feature gaps are not permanent - GitHub is working to close them. You can review [our public roadmap](https://github.com/github/roadmap) for more details on when these scenarios will be supported.

The major gaps in fine-grained personal access tokens are:

* Using fine-grained personal access token to contribute to public repos where the user is not a member.
* Using fine-grained personal access token to contribute to repositories where the user is an outside or repository collaborator.
* Using fine-grained personal access token to access multiple organizations at once.
* Using fine-grained personal access token to access Packages.
* Using fine-grained personal access token to call the Checks API.
* Using fine-grained personal access token to access Projects owned by a user account.

All of these gaps will be solved over time, as GitHub continues to invest in more secure access patterns.

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.
* Outside collaborators can only use personal access tokens (classic) to access organization repositories that they are a collaborator on.
* A few REST API endpoints are only available with a personal access tokens (classic). To check whether an endpoint also supports fine-grained personal access tokens, see the documentation for that endpoint, or see [Endpoints available for fine-grained personal access tokens](/en/rest/overview/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.

As a security precaution, GitHub automatically removes personal access tokens that haven't been used in a year. To provide additional security, we highly recommend adding an expiration to your personal access tokens.

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](/en/github-cli/github-cli/about-github-cli) or [Git Credential Manager](https://github.com/GitCredentialManager/git-credential-manager/blob/main/README.md) 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 [Use GITHUB\_TOKEN for authentication in workflows](/en/actions/security-guides/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](/en/actions/security-guides/encrypted-secrets). You can also store your token as a Codespaces secret and run your script in Codespaces. For more information, see [Managing your account-specific secrets for GitHub Codespaces](/en/codespaces/managing-your-codespaces/managing-encrypted-secrets-for-your-codespaces).

For more information about best practices, see [Keeping your API credentials secure](/en/rest/overview/keeping-your-api-credentials-secure).

Creating a fine-grained personal access token

> \[!NOTE]
> There is a limit of 50 fine-grained personal access tokens you can create. If you require more tokens or are building automations, consider using a GitHub App for better scalability and management. For more information, see [Deciding when to build a GitHub App](/en/apps/creating-github-apps/about-creating-github-apps/deciding-when-to-build-a-github-app#choosing-between-a-github-app-or-a-personal-access-token).

1. [Verify your email address](/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-email-preferences/verifying-your-email-address), if it hasn't been verified yet.

2. In the upper-right corner of any page on GitHub, click your profile picture, then click **<svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-gear" aria-label="gear" role="img"><path d="M8 0a8.2 8.2 0 0 1 .701.031C9.444.095 9.99.645 10.16 1.29l.288 1.107c.018.066.079.158.212.224.231.114.454.243.668.386.123.082.233.09.299.071l1.103-.303c.644-.176 1.392.021 1.82.63.27.385.506.792.704 1.218.315.675.111 1.422-.364 1.891l-.814.806c-.049.048-.098.147-.088.294.016.257.016.515 0 .772-.01.147.038.246.088.294l.814.806c.475.469.679 1.216.364 1.891a7.977 7.977 0 0 1-.704 1.217c-.428.61-1.176.807-1.82.63l-1.102-.302c-.067-.019-.177-.011-.3.071a5.909 5.909 0 0 1-.668.386c-.133.066-.194.158-.211.224l-.29 1.106c-.168.646-.715 1.196-1.458 1.26a8.006 8.006 0 0 1-1.402 0c-.743-.064-1.289-.614-1.458-1.26l-.289-1.106c-.018-.066-.079-.158-.212-.224a5.738 5.738 0 0 1-.668-.386c-.123-.082-.233-.09-.299-.071l-1.103.303c-.644.176-1.392-.021-1.82-.63a8.12 8.12 0 0 1-.704-1.218c-.315-.675-.111-1.422.363-1.891l.815-.806c.05-.048.098-.147.088-.294a6.214 6.214 0 0 1 0-.772c.01-.147-.038-.246-.088-.294l-.815-.806C.635 6.045.431 5.298.746 4.623a7.92 7.92 0 0 1 .704-1.217c.428-.61 1.176-.807 1.82-.63l1.102.302c.067.019.177.011.3-.071.214-.143.437-.272.668-.386.133-.066.194-.158.211-.224l.29-1.106C6.009.645 6.556.095 7.299.03 7.53.01 7.764 0 8 0Zm-.571 1.525c-.036.003-.108.036-.137.146l-.289 1.105c-.147.561-.549.967-.998 1.189-.173.086-.34.183-.5.29-.417.278-.97.423-1.529.27l-1.103-.303c-.109-.03-.175.016-.195.045-.22.312-.412.644-.573.99-.014.031-.021.11.059.19l.815.806c.411.406.562.957.53 1.456a4.709 4.709 0 0 0 0 .582c.032.499-.119 1.05-.53 1.456l-.815.806c-.081.08-.073.159-.059.19.162.346.353.677.573.989.02.03.085.076.195.046l1.102-.303c.56-.153 1.113-.008 1.53.27.161.107.328.204.501.29.447.222.85.629.997 1.189l.289 1.105c.029.109.101.143.137.146a6.6 6.6 0 0 0 1.142 0c.036-.003.108-.036.137-.146l.289-1.105c.147-.561.549-.967.998-1.189.173-.086.34-.183.5-.29.417-.278.97-.423 1.529-.27l1.103.303c.109.029.175-.016.195-.045.22-.313.411-.644.573-.99.014-.031.021-.11-.059-.19l-.815-.806c-.411-.406-.562-.957-.53-1.456a4.709 4.709 0 0 0 0-.582c-.032-.499.119-1.05.53-1.456l.815-.806c.081-.08.073-.159.059-.19a6.464 6.464 0 0 0-.573-.989c-.02-.03-.085-.076-.195-.046l-1.102.303c-.56.153-1.113.008-1.53-.27a4.44 4.44 0 0 0-.501-.29c-.447-.222-.85-.629-.997-1.189l-.289-1.105c-.029-.11-.101-.143-.137-.146a6.6 6.6 0 0 0-1.142 0ZM11 8a3 3 0 1 1-6 0 3 3 0 0 1 6 0ZM9.5 8a1.5 1.5 0 1 0-3.001.001A1.5 1.5 0 0 0 9.5 8Z"></path></svg> Settings**.

3. In the left sidebar, click **<svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-code" aria-label="code" role="img"><path d="m11.28 3.22 4.25 4.25a.75.75 0 0 1 0 1.06l-4.25 4.25a.749.749 0 0 1-1.275-.326.749.749 0 0 1 .215-.734L13.94 8l-3.72-3.72a.749.749 0 0 1 .326-1.275.749.749 0 0 1 .734.215Zm-6.56 0a.751.751 0 0 1 1.042.018.751.751 0 0 1 .018 1.042L2.06 8l3.72 3.72a.749.749 0 0 1-.326 1.275.749.749 0 0 1-.734-.215L.47 8.53a.75.75 0 0 1 0-1.06Z"></path></svg> Developer settings**.

4. In the left sidebar, under **<svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-key" aria-label="key" role="img"><path d="M10.5 0a5.499 5.499 0 1 1-1.288 10.848l-.932.932a.749.749 0 0 1-.53.22H7v.75a.749.749 0 0 1-.22.53l-.5.5a.749.749 0 0 1-.53.22H5v.75a.749.749 0 0 1-.22.53l-.5.5a.749.749 0 0 1-.53.22h-2A1.75 1.75 0 0 1 0 14.25v-2c0-.199.079-.389.22-.53l4.932-4.932A5.5 5.5 0 0 1 10.5 0Zm-4 5.5c-.001.431.069.86.205 1.269a.75.75 0 0 1-.181.768L1.5 12.56v1.69c0 .138.112.25.25.25h1.69l.06-.06v-1.19a.75.75 0 0 1 .75-.75h1.19l.06-.06v-1.19a.75.75 0 0 1 .75-.75h1.19l1.023-1.025a.75.75 0 0 1 .768-.18A4 4 0 1 0 6.5 5.5ZM11 6a1 1 0 1 1 0 2 1 1 0 0 1 0-2Z"></path></svg> Personal access tokens**, click **Fine-grained tokens**.

5. Click **Generate new token**.

6. Under **Token name**, enter a name for the token.

7. Under **Expiration**, select an expiration for the token. Infinite lifetimes are allowed but may be blocked by a maximum lifetime policy set by your organization or enterprise owner. For more information, See [Enforcing a maximum lifetime policy for personal access tokens](/en/organizations/managing-programmatic-access-to-your-organization/setting-a-personal-access-token-policy-for-your-organization#enforcing-a-maximum-lifetime-policy-for-personal-access-tokens).

8. Optionally, under **Description**, add a note to describe the purpose of the token.

9. 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 if the organization has blocked the use of fine-grained personal access tokens. For more information, see [Setting a personal access token policy for your organization](/en/organizations/managing-programmatic-access-to-your-organization/setting-a-personal-access-token-policy-for-your-organization).

10. 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.

11. 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.

12. 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.

13. 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.

The REST API reference document for each endpoint states whether the endpoint works with fine-grained personal access tokens and states what permissions are required in order for the token to use the endpoint. Some endpoints may require multiple permissions, and some endpoints may require one of multiple permissions. For an overview of which REST API endpoints a fine-grained personal access token can access with each permission, see [Permissions required for fine-grained personal access tokens](/en/rest/overview/permissions-required-for-fine-grained-personal-access-tokens).

14. 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](/en/organizations/managing-programmatic-access-to-your-organization/reviewing-and-revoking-personal-access-tokens-in-your-organization).

Pre-filling fine-grained personal access token details using URL parameters

You can share templates for a fine-grained personal access token via links. Storing token details this way makes it easier to automate workflows and improve your developer experience by directing users to token creation with relevant fields already completed.

Each supported field can be set using a specific query parameter. All parameters are optional and validated by the token generation form to ensure that the combinations of permissions and resource owner makes sense.

An example URL template is shown here, with line breaks for legibility:

```http copy
https://github.com/settings/personal-access-tokens/new
  ?name=Repo-reading+token
  &description=Just+contents:read
  &target_name=octodemo
  &expires_in=45
  &contents=read
```

Try the URL to create a token with `contents:read` and `metadata:read`, with the given name and description and an expiration date 45 days in the future. You'll see an error message indicating `Cannot find the specified resource owner: octodemo` because you're not a member of the `octodemo` organization.

Below are some example URLs that generate the tokens we see most often:

* [Read repo contents](https://github.com/settings/personal-access-tokens/new?name=Repo-reading+token\&description=Just+contents:read\&contents=read)
* [Push access to repos](https://github.com/settings/personal-access-tokens/new?name=Repo-writing+token\&description=Just+contents:write\&contents=write)
* [GitHub Models access](https://github.com/settings/personal-access-tokens/new?name=GitHub+Models+token\&description=Used%20to%20call%20GitHub%20Models%20APIs%20to%20easily%20run%20LLMs%3A%20https%3A%2F%2Fdocs.github.com%2Fgithub-models%2Fquickstart%23step-2-make-an-api-call\&user_models=read)<!-- markdownlint-disable-line search-replace Custom rule -->
* [Update code and open a PR](https://github.com/settings/personal-access-tokens/new?name=Core-loop+token\&description=Write%20code%20and%20push%20it%20to%20main%21%20Includes%20permission%20to%20edit%20workflow%20files%20for%20Actions%20-%20remove%20%60workflows%3Awrite%60%20if%20you%20don%27t%20need%20to%20do%20that\&contents=write\&pull_requests=write\&workflows=write)
* [Manage Copilot licenses in an organization](https://github.com/settings/personal-access-tokens/new?name=Core-loop+token\&description=Enable%20or%20disable%20copilot%20access%20for%20users%20with%20the%20Seat%20Management%20APIs%3A%20https%3A%2F%2Fdocs.github.com%2Frest%2Fcopilot%2Fcopilot-user-management%0ABe%20sure%20to%20select%20an%20organization%20for%20your%20resource%20owner%20below%21\&organization_copilot_seat_management=write)<!-- markdownlint-disable-line search-replace Custom rule -->
* [Make Copilot requests](https://github.com/settings/personal-access-tokens/new?name=Copilot+requests+token\&description=Make%20Copilot%20API%20requests%20on%20behalf%20of%20the%20user%2C%20consuming%20premium%20requests%3A%20https%3A%2F%2Fdocs.github.com%2Fcopilot%2Fconcepts%2Fbilling%2Fcopilot-requests\&copilot_requests=write)<!-- markdownlint-disable-line search-replace Custom rule -->

Supported Query Parameters

To create your own token template, follow the query parameter details provided in this table:

Parameter	Type	Example Value	Valid Values	Description
`name`	string	`Deploy%20Bot`	≤ 40 characters, URL-encoded	Pre-fills the token’s display name.
`description`	string	`Used+for+deployments`	≤ 1024 chars, URL-encoded	Pre-fills the description for the token.
`target_name`	string	`octodemo`	User or organization slug	Sets the token's resource target. This is the owner of the repositories that the token will be able to access. If not provided, defaults to the current user's account.
`expires_in`	integer	`30` or `none`	Integer between 1 and 366, or `none`	Days until expiration or `none` for non-expiring. If not provided, the default is 30 days, or less if the target has a token lifetime policy set.
`<permission>`	string	`contents=read`	A series of permission and access levels.	The permissions the token should have. Permissions can be set to `read`, `write`, or `admin`, but not every permission supports each of those levels.

Permissions

Each supported permission is set using its name as a query parameter, with the value specifying the desired access level. Valid access levels are `read`, `write`, and `admin`. Some permissions only support `read`, some only support `write`, and only a few have `admin`. Use as many permissions as needed, in the form `&contents=read&pull_requests=write&...`.

You do not need to include both `read` and `write` for a permission in your URL—`write` always includes `read`, and `admin` always includes `write`.

Account Permissions

Account permissions are only used when the current user is set as the resource owner.

Parameter name	Display name	Access levels
`blocking`	Block another user	`read`, `write`
`codespaces_user_secrets`	Codespaces user secrets	`read`, `write`
`copilot_messages`	Copilot Chat	`read`
`copilot_editor_context`	Copilot Editor Context	`read`
`copilot_requests`	Copilot requests	`write`
`emails`	Email addresses	`read`, `write`
`user_events`	Events	`read`
`followers`	Followers	`read`, `write`
`gpg_keys`	GPG keys	`read`, `write`
`gists`	Gists	`write`
`keys`	Git SSH keys	`read`, `write`
`interaction_limits`	Interaction limits	`read`, `write`
`knowledge_bases`	Knowledge bases	`read`, `write`
`user_models`	Models	`read`
`plan`	Plan	`read`
`private_repository_invitations`	Private repository invitations	`read`
`profile`	Profile	`write`
`git_signing_ssh_public_keys`	SSH signing keys	`read`, `write`
`starring`	Starring	`read`, `write`
`watching`	Watching	`read`, `write`

> \[!NOTE]
> The `copilot_requests` permission enables making Copilot requests for the given user, which count towards the user's premium request allowance or are charged to overage billing if the allowance is exceeded. For more information about Copilot requests and billing, see [Requests in GitHub Copilot](/en/copilot/concepts/billing/copilot-requests).

Repository Permissions

Repository permissions work for both user and organization resource owners.

Parameter name	Display name	Access levels
`actions`	Actions	`read`, `write`
`administration`	Administration	`read`, `write`
`artifact_metadata`	Artifact Metadata	`read`, `write`
`attestations`	Attestations	`read`, `write`
`security_events`	Code scanning alerts	`read`, `write`
`codespaces`	Codespaces	`read`, `write`
`codespaces_lifecycle_admin`	Codespaces lifecycle admin	`read`, `write`
`codespaces_metadata`	Codespaces metadata	`read`
`codespaces_secrets`	Codespaces secrets	`write`
`statuses`	Commit statuses	`read`, `write`
`contents`	Contents	`read`, `write`
`repository_custom_properties`	Custom properties	`read`, `write`
`vulnerability_alerts`	Dependabot alerts	`read`, `write`
`dependabot_secrets`	Dependabot secrets	`read`, `write`
`deployments`	Deployments	`read`, `write`
`discussions`	Discussions	`read`, `write`
`environments`	Environments	`read`, `write`
`issues`	Issues	`read`, `write`
`merge_queues`	Merge queues	`read`, `write`
`metadata`	Metadata	`read`
`pages`	Pages	`read`, `write`
`pull_requests`	Pull requests	`read`, `write`
`repository_advisories`	Repository security advisories	`read`, `write`
`secret_scanning_alerts`	Secret scanning alerts	`read`, `write`
`secrets`	Secrets	`read`, `write`
`actions_variables`	Variables	`read`, `write`
`repository_hooks`	Webhooks	`read`, `write`
`workflows`	Workflows	`write`

Organization Permissions

Organization permissions can only be used if the resource owner is an organization.

Parameter name	Display name	Access levels
`organization_api_insights`	API Insights	`read`
`organization_administration`	Administration	`read`, `write`
`organization_user_blocking`	Blocking users	`read`, `write`
`organization_campaigns`	Campaigns	`read`, `write`
`organization_custom_org_roles`	Custom organization roles	`read`, `write`
`organization_custom_properties`	Custom repository properties	`read`, `write`, `admin`
`organization_custom_roles`	Custom repository roles	`read`, `write`
`organization_events`	Events	`read`
`organization_copilot_seat_management`	GitHub Copilot Business	`read`, `write`
`issue_types`	Issue Types	`read`, `write`
`organization_knowledge_bases`	Knowledge bases	`read`, `write`
`members`	Members	`read`, `write`
`organization_models`	Models	`read`
`organization_network_configurations`	Network configurations	`read`, `write`
`organization_announcement_banners`	Organization announcement banners	`read`, `write`
`organization_codespaces`	Organization codespaces	`read`, `write`
`organization_codespaces_secrets`	Organization codespaces secrets	`read`, `write`
`organization_codespaces_settings`	Organization codespaces settings	`read`, `write`
`organization_dependabot_secrets`	Organization dependabot secrets	`read`, `write`
`organization_code_scanning_dismissal_requests`	Code scanning dismissal requests	`read`, `write`
`organization_private_registries`	Private registries	`read`, `write`
`organization_plan`	Plan	`read`
`organization_projects`	Projects	`read`, `write`, `admin`
`organization_secrets`	Secrets	`read`, `write`
`organization_self_hosted_runners`	Self-hosted runners	`read`, `write`
`team_discussions`	Team discussions	`read`, `write`
`organization_actions_variables`	Variables	`read`, `write`
`organization_hooks`	Webhooks	`read`, `write`

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.

> \[!WARNING]
> 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. [Verify your email address](/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-email-preferences/verifying-your-email-address), if it hasn't been verified yet.
2. In the upper-right corner of any page on GitHub, click your profile picture, then click **<svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-gear" aria-label="gear" role="img"><path d="M8 0a8.2 8.2 0 0 1 .701.031C9.444.095 9.99.645 10.16 1.29l.288 1.107c.018.066.079.158.212.224.231.114.454.243.668.386.123.082.233.09.299.071l1.103-.303c.644-.176 1.392.021 1.82.63.27.385.506.792.704 1.218.315.675.111 1.422-.364 1.891l-.814.806c-.049.048-.098.147-.088.294.016.257.016.515 0 .772-.01.147.038.246.088.294l.814.806c.475.469.679 1.216.364 1.891a7.977 7.977 0 0 1-.704 1.217c-.428.61-1.176.807-1.82.63l-1.102-.302c-.067-.019-.177-.011-.3.071a5.909 5.909 0 0 1-.668.386c-.133.066-.194.158-.211.224l-.29 1.106c-.168.646-.715 1.196-1.458 1.26a8.006 8.006 0 0 1-1.402 0c-.743-.064-1.289-.614-1.458-1.26l-.289-1.106c-.018-.066-.079-.158-.212-.224a5.738 5.738 0 0 1-.668-.386c-.123-.082-.233-.09-.299-.071l-1.103.303c-.644.176-1.392-.021-1.82-.63a8.12 8.12 0 0 1-.704-1.218c-.315-.675-.111-1.422.363-1.891l.815-.806c.05-.048.098-.147.088-.294a6.214 6.214 0 0 1 0-.772c.01-.147-.038-.246-.088-.294l-.815-.806C.635 6.045.431 5.298.746 4.623a7.92 7.92 0 0 1 .704-1.217c.428-.61 1.176-.807 1.82-.63l1.102.302c.067.019.177.011.3-.071.214-.143.437-.272.668-.386.133-.066.194-.158.211-.224l.29-1.106C6.009.645 6.556.095 7.299.03 7.53.01 7.764 0 8 0Zm-.571 1.525c-.036.003-.108.036-.137.146l-.289 1.105c-.147.561-.549.967-.998 1.189-.173.086-.34.183-.5.29-.417.278-.97.423-1.529.27l-1.103-.303c-.109-.03-.175.016-.195.045-.22.312-.412.644-.573.99-.014.031-.021.11.059.19l.815.806c.411.406.562.957.53 1.456a4.709 4.709 0 0 0 0 .582c.032.499-.119 1.05-.53 1.456l-.815.806c-.081.08-.073.159-.059.19.162.346.353.677.573.989.02.03.085.076.195.046l1.102-.303c.56-.153 1.113-.008 1.53.27.161.107.328.204.501.29.447.222.85.629.997 1.189l.289 1.105c.029.109.101.143.137.146a6.6 6.6 0 0 0 1.142 0c.036-.003.108-.036.137-.146l.289-1.105c.147-.561.549-.967.998-1.189.173-.086.34-.183.5-.29.417-.278.97-.423 1.529-.27l1.103.303c.109.029.175-.016.195-.045.22-.313.411-.644.573-.99.014-.031.021-.11-.059-.19l-.815-.806c-.411-.406-.562-.957-.53-1.456a4.709 4.709 0 0 0 0-.582c-.032-.499.119-1.05.53-1.456l.815-.806c.081-.08.073-.159.059-.19a6.464 6.464 0 0 0-.573-.989c-.02-.03-.085-.076-.195-.046l-1.102.303c-.56.153-1.113.008-1.53-.27a4.44 4.44 0 0 0-.501-.29c-.447-.222-.85-.629-.997-1.189l-.289-1.105c-.029-.11-.101-.143-.137-.146a6.6 6.6 0 0 0-1.142 0ZM11 8a3 3 0 1 1-6 0 3 3 0 0 1 6 0ZM9.5 8a1.5 1.5 0 1 0-3.001.001A1.5 1.5 0 0 0 9.5 8Z"></path></svg> Settings**.
3. In the left sidebar, click **<svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-code" aria-label="code" role="img"><path d="m11.28 3.22 4.25 4.25a.75.75 0 0 1 0 1.06l-4.25 4.25a.749.749 0 0 1-1.275-.326.749.749 0 0 1 .215-.734L13.94 8l-3.72-3.72a.749.749 0 0 1 .326-1.275.749.749 0 0 1 .734.215Zm-6.56 0a.751.751 0 0 1 1.042.018.751.751 0 0 1 .018 1.042L2.06 8l3.72 3.72a.749.749 0 0 1-.326 1.275.749.749 0 0 1-.734-.215L.47 8.53a.75.75 0 0 1 0-1.06Z"></path></svg> Developer settings**.
4. In the left sidebar, under **<svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-key" aria-label="key" role="img"><path d="M10.5 0a5.499 5.499 0 1 1-1.288 10.848l-.932.932a.749.749 0 0 1-.53.22H7v.75a.749.749 0 0 1-.22.53l-.5.5a.749.749 0 0 1-.53.22H5v.75a.749.749 0 0 1-.22.53l-.5.5a.749.749 0 0 1-.53.22h-2A1.75 1.75 0 0 1 0 14.25v-2c0-.199.079-.389.22-.53l4.932-4.932A5.5 5.5 0 0 1 10.5 0Zm-4 5.5c-.001.431.069.86.205 1.269a.75.75 0 0 1-.181.768L1.5 12.56v1.69c0 .138.112.25.25.25h1.69l.06-.06v-1.19a.75.75 0 0 1 .75-.75h1.19l.06-.06v-1.19a.75.75 0 0 1 .75-.75h1.19l1.023-1.025a.75.75 0 0 1 .768-.18A4 4 0 1 0 6.5 5.5ZM11 6a1 1 0 1 1 0 2 1 1 0 0 1 0-2Z"></path></svg> Personal access tokens**, click **Tokens (classic)**.
5. Select **Generate new token**, then click **Generate new token (classic)**.
6. In the "Note" field, give your token a descriptive name.
7. To give your token an expiration, select **Expiration**, then choose a default option or click **Custom** to enter a date.
8. 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](/en/apps/oauth-apps/building-oauth-apps/scopes-for-oauth-apps#available-scopes).
9. Click **Generate token**.
10. Optionally, to copy the new token to your clipboard, click <svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-copy" aria-label="Copy token" role="img"><path d="M0 6.75C0 5.784.784 5 1.75 5h1.5a.75.75 0 0 1 0 1.5h-1.5a.25.25 0 0 0-.25.25v7.5c0 .138.112.25.25.25h7.5a.25.25 0 0 0 .25-.25v-1.5a.75.75 0 0 1 1.5 0v1.5A1.75 1.75 0 0 1 9.25 16h-7.5A1.75 1.75 0 0 1 0 14.25Z"></path><path d="M5 1.75C5 .784 5.784 0 6.75 0h7.5C15.216 0 16 .784 16 1.75v7.5A1.75 1.75 0 0 1 14.25 11h-7.5A1.75 1.75 0 0 1 5 9.25Zm1.75-.25a.25.25 0 0 0-.25.25v7.5c0 .138.112.25.25.25h7.5a.25.25 0 0 0 .25-.25v-7.5a.25.25 0 0 0-.25-.25Z"></path></svg>.

   ![Screenshot of the "Personal access tokens" page. Next to a blurred-out token, an icon of two overlapping squares is outlined in orange.](/assets/images/help/settings/personal-access-tokens.png)
11. To use your token to access resources owned by an organization that uses SAML single sign-on, authorize the token. For more information, see [Authorizing a personal access token for use with single sign-on](/en/enterprise-cloud@latest/authentication/authenticating-with-saml-single-sign-on/authorizing-a-personal-access-token-for-use-with-saml-single-sign-on) in the GitHub Enterprise Cloud documentation.

Deleting a personal access token

You should delete a personal access token if it is no longer needed. If you delete a personal access token that was used to create a deploy key, the deploy key will also be deleted.

1. In the upper-right corner of any page on GitHub, click your profile picture, then click **<svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-gear" aria-label="gear" role="img"><path d="M8 0a8.2 8.2 0 0 1 .701.031C9.444.095 9.99.645 10.16 1.29l.288 1.107c.018.066.079.158.212.224.231.114.454.243.668.386.123.082.233.09.299.071l1.103-.303c.644-.176 1.392.021 1.82.63.27.385.506.792.704 1.218.315.675.111 1.422-.364 1.891l-.814.806c-.049.048-.098.147-.088.294.016.257.016.515 0 .772-.01.147.038.246.088.294l.814.806c.475.469.679 1.216.364 1.891a7.977 7.977 0 0 1-.704 1.217c-.428.61-1.176.807-1.82.63l-1.102-.302c-.067-.019-.177-.011-.3.071a5.909 5.909 0 0 1-.668.386c-.133.066-.194.158-.211.224l-.29 1.106c-.168.646-.715 1.196-1.458 1.26a8.006 8.006 0 0 1-1.402 0c-.743-.064-1.289-.614-1.458-1.26l-.289-1.106c-.018-.066-.079-.158-.212-.224a5.738 5.738 0 0 1-.668-.386c-.123-.082-.233-.09-.299-.071l-1.103.303c-.644.176-1.392-.021-1.82-.63a8.12 8.12 0 0 1-.704-1.218c-.315-.675-.111-1.422.363-1.891l.815-.806c.05-.048.098-.147.088-.294a6.214 6.214 0 0 1 0-.772c.01-.147-.038-.246-.088-.294l-.815-.806C.635 6.045.431 5.298.746 4.623a7.92 7.92 0 0 1 .704-1.217c.428-.61 1.176-.807 1.82-.63l1.102.302c.067.019.177.011.3-.071.214-.143.437-.272.668-.386.133-.066.194-.158.211-.224l.29-1.106C6.009.645 6.556.095 7.299.03 7.53.01 7.764 0 8 0Zm-.571 1.525c-.036.003-.108.036-.137.146l-.289 1.105c-.147.561-.549.967-.998 1.189-.173.086-.34.183-.5.29-.417.278-.97.423-1.529.27l-1.103-.303c-.109-.03-.175.016-.195.045-.22.312-.412.644-.573.99-.014.031-.021.11.059.19l.815.806c.411.406.562.957.53 1.456a4.709 4.709 0 0 0 0 .582c.032.499-.119 1.05-.53 1.456l-.815.806c-.081.08-.073.159-.059.19.162.346.353.677.573.989.02.03.085.076.195.046l1.102-.303c.56-.153 1.113-.008 1.53.27.161.107.328.204.501.29.447.222.85.629.997 1.189l.289 1.105c.029.109.101.143.137.146a6.6 6.6 0 0 0 1.142 0c.036-.003.108-.036.137-.146l.289-1.105c.147-.561.549-.967.998-1.189.173-.086.34-.183.5-.29.417-.278.97-.423 1.529-.27l1.103.303c.109.029.175-.016.195-.045.22-.313.411-.644.573-.99.014-.031.021-.11-.059-.19l-.815-.806c-.411-.406-.562-.957-.53-1.456a4.709 4.709 0 0 0 0-.582c-.032-.499.119-1.05.53-1.456l.815-.806c.081-.08.073-.159.059-.19a6.464 6.464 0 0 0-.573-.989c-.02-.03-.085-.076-.195-.046l-1.102.303c-.56.153-1.113.008-1.53-.27a4.44 4.44 0 0 0-.501-.29c-.447-.222-.85-.629-.997-1.189l-.289-1.105c-.029-.11-.101-.143-.137-.146a6.6 6.6 0 0 0-1.142 0ZM11 8a3 3 0 1 1-6 0 3 3 0 0 1 6 0ZM9.5 8a1.5 1.5 0 1 0-3.001.001A1.5 1.5 0 0 0 9.5 8Z"></path></svg> Settings**.
2. In the left sidebar, click **<svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-code" aria-label="code" role="img"><path d="m11.28 3.22 4.25 4.25a.75.75 0 0 1 0 1.06l-4.25 4.25a.749.749 0 0 1-1.275-.326.749.749 0 0 1 .215-.734L13.94 8l-3.72-3.72a.749.749 0 0 1 .326-1.275.749.749 0 0 1 .734.215Zm-6.56 0a.751.751 0 0 1 1.042.018.751.751 0 0 1 .018 1.042L2.06 8l3.72 3.72a.749.749 0 0 1-.326 1.275.749.749 0 0 1-.734-.215L.47 8.53a.75.75 0 0 1 0-1.06Z"></path></svg> Developer settings**.
3. In the left sidebar, under **<svg version="1.1" width="16" height="16" viewBox="0 0 16 16" class="octicon octicon-key" aria-label="key" role="img"><path d="M10.5 0a5.499 5.499 0 1 1-1.288 10.848l-.932.932a.749.749 0 0 1-.53.22H7v.75a.749.749 0 0 1-.22.53l-.5.5a.749.749 0 0 1-.53.22H5v.75a.749.749 0 0 1-.22.53l-.5.5a.749.749 0 0 1-.53.22h-2A1.75 1.75 0 0 1 0 14.25v-2c0-.199.079-.389.22-.53l4.932-4.932A5.5 5.5 0 0 1 10.5 0Zm-4 5.5c-.001.431.069.86.205 1.269a.75.75 0 0 1-.181.768L1.5 12.56v1.69c0 .138.112.25.25.25h1.69l.06-.06v-1.19a.75.75 0 0 1 .75-.75h1.19l.06-.06v-1.19a.75.75 0 0 1 .75-.75h1.19l1.023-1.025a.75.75 0 0 1 .768-.18A4 4 0 1 0 6.5 5.5ZM11 6a1 1 0 1 1 0 2 1 1 0 0 1 0-2Z"></path></svg> 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**.

> \[!NOTE] If you find a leaked personal access token belonging to someone else, you can submit a revocation request through the REST API. See [Best practices for preventing data leaks in your organization](/en/code-security/getting-started/best-practices-for-preventing-data-leaks-in-your-organization#mitigate-data-leaks).

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.

```shell
$ git clone https://github.com/USERNAME/REPO.git
Username: YOUR-USERNAME
Password: YOUR-PERSONAL-ACCESS-TOKEN
```

Although you are required to enter your username along with your personal access token, the username is not used to authenticate you. Instead, the personal access token is used to authenticate you. If you do not enter a username, you will receive an error message that your credentials are invalid.

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](/en/get-started/git-basics/managing-remote-repositories#switching-remote-urls-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](/en/get-started/git-basics/updating-credentials-from-the-macos-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](/en/get-started/git-basics/caching-your-github-credentials-in-git).

Further reading

* [About authentication to GitHub](/en/authentication/keeping-your-account-and-data-secure/about-authentication-to-github)
* [Token expiration and revocation](/en/authentication/keeping-your-account-and-data-secure/token-expiration-and-revocation)