Keeping your API credentials secure

In this article

Follow these best practices to keep your API credentials and tokens secure.

Choose an appropriate authentication method

You should choose an authentication method that is appropriate for the task you want to accomplish.

- To use the API for personal use, you can create a personal access token.
- To use the API on behalf of an organization or another user, you should create a GitHub App.
- To use the API in a GitHub Actions workflow, you should authenticate with the built-in `GITHUB_TOKEN`.

For more information, see "About authentication to GitHub."

Limit the permissions of your credentials

When creating a personal access token, only select the minimum permissions or scopes needed, and set an expiration date for the minimum amount of time you'll need to use the token. GitHub recommends that you use fine-grained personal access tokens instead of personal access tokens (classic). For more information, see "Managing your personal access tokens."

When creating a GitHub App, select the minimum permissions that your GitHub App will need. For more information, see "Best practices for creating a GitHub App."

When authenticating with `GITHUB_TOKEN` in a GitHub Actions workflow, only give the minimum amount of permissions needed. For more information, see "Automatic token authentication."

Store your authentication credentials securely

Treat authentication credentials the same way you would treat your passwords or other sensitive credentials.

- Don't share authentication credentials using an unencrypted messaging or email system.

- Don't pass your personal access token as plain text in the command line. For more information, see "Managing your personal access tokens."
- Don't push unencrypted authentication credentials like tokens or keys to any repository, even if the repository is private. Instead consider using a GitHub Actions secret or Codespaces secret. For more information, see "Using secrets in GitHub Actions" and "Managing secrets for your codespaces".
- You can use secret scanning to discover tokens, private keys, and other secrets that were pushed to a repository, or to block future pushes that contain secrets. For more information, see "About secret scanning."

Limit who can access your authentication credentials

Don't share your personal access token with others. Instead of sharing a personal access token, consider creating a GitHub App. For more information, see "About creating GitHub Apps."

If you need to share credentials with a team, store the credentials in a secure shared system. For example, you could store and share passwords securely using 1Password, or you could store keys in Azure KeyVault and manage access with your IAM (Identity and access management).

If you're creating a GitHub Actions workflow that needs to access the API, you can store your credentials in an encrypted secret, and access the encrypted secret from the workflow. For more information, see "Using secrets in GitHub Actions" and "Making authenticated API requests with a GitHub App in a GitHub Actions workflow".

Use authentication credentials securely in your code

Never hardcode authentication credentials like tokens, keys, or app-related secrets into your code. Instead, consider using a secret manager such as Azure Key Vault or HashiCorp Vault. For more information about securing GitHub App credentials, see "Best practices for creating a GitHub App."

When using a personal access token in a script, consider storing your token as a GitHub Actions secret and running your script through GitHub Actions. You can also store your token as a Codespaces secret and run your script in Codespaces. For more information, see "Using secrets in GitHub Actions" and "Managing secrets for your codespaces".

If none of these options are possible, you can store authentication credentials in a `.env` file. Make sure to encrypt your `.env` file, and never push it to any repository.

Prepare a remediation plan

You should create a plan to handle any security breaches in a timely manner. In the event that your token or other authentication credential is leaked, you will need to:

- Generate a new credential.
- Replace the old credential with the new one everywhere that you are storing or accessing the credential.
- Delete the old compromised credential.

For information about rotating compromised credentials for a GitHub App, see "Best practices for creating a GitHub App."

For information about creating and deleting personal access tokens, see "Managing your personal access tokens."