Managing private keys for GitHub Apps

In this article

You can manage private keys to authenticate with your GitHub App.

About private keys for GitHub Apps

After you create a GitHub App, you'll need to generate a private key in order to make requests to the GitHub API as the application itself. For example, you need a private key to sign a JSON Web Token (JWT) in order to request an installation access token. For more information, see "Generating a JSON Web Token (JWT) for a GitHub App"

You can create multiple private keys and rotate them to prevent downtime if a key is compromised or lost. To verify that a private key matches a public key, see "Verifying private keys".

Private keys do not expire and instead need to be manually revoked. For more information about how to revoke a private key, see "Deleting private keys."

You must keep private keys for GitHub Apps secure. For more information, see "Storing private keys".

Generating private keys

To generate a private key:

1. In the upper-right corner of any page on GitHub, click your profile photo.
2. Navigate to your account settings.
   - For a GitHub App owned by a personal account, click **Settings**.
   - For a GitHub App owned by an organization:
     a. Click **Your organizations**.
     b. To the right of the organization, click **Settings**.
3. In the left sidebar, click <> **Developer settings**.
4. In the left sidebar, click **GitHub Apps**.

5. Next to the GitHub App that you want to generate a private key for, click **Edit**.
6. Under "Private keys", click **Generate a private key**.
7. You will see a private key in PEM format downloaded to your computer. Make sure to store this file because GitHub only stores the public portion of the key. For more information about securely storing your key, see "Storing private keys."

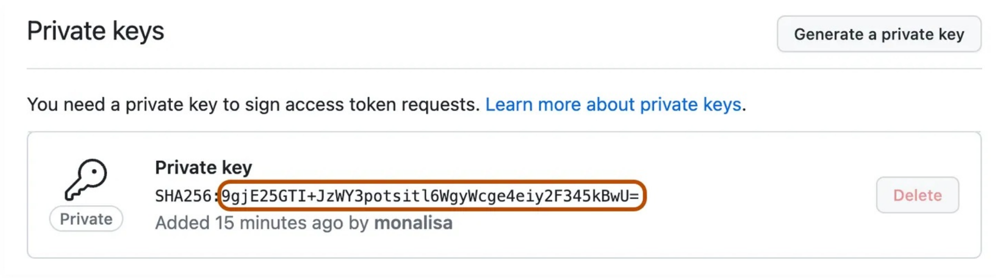
> **Note:** If you're using a library that requires a specific file format, the PEM file you download will be in `PKCS#1 RSAPrivateKey` format.

Verifying private keys

GitHub generates a fingerprint for each private and public key pair using the SHA-256 hash function. You can verify that your private key matches the public key stored on GitHub by generating the fingerprint of your private key and comparing it to the fingerprint shown on GitHub.

To verify a private key:

1. Find the fingerprint for the private and public key pair you want to verify in the "Private keys" section of the settings page for your GitHub App. For more information, see "Generating private keys".



You need a private key to sign access token requests. Learn more about private keys.

Private key
SHA256:9gjE25GTI+JzWY3potsitl6WgyWcge4eiy2F345kBwU=
Added 15 minutes ago by **monalisa**

2. Generate the fingerprint of your private key (PEM) locally by using the following command:

```
openssl rsa -in PATH_TO_PEM_FILE -pubout -outform DER | openssl sha256 -
binary | openssl base64
```

3. Compare the results of the locally generated fingerprint to the fingerprint you see in GitHub.

Deleting private keys

You can remove a lost or compromised private key by deleting it, but you must regenerate a new key before you can delete the existing key.

1. In the upper-right corner of any page on GitHub, click your profile photo.
2. Navigate to your account settings.
   - For a GitHub App owned by a personal account, click **Settings**.
   - For a GitHub App owned by an organization:

   a. Click **Your organizations**.

   b. To the right of the organization, click **Settings**.

3. In the left sidebar, click <> **Developer settings**.
4. In the left sidebar, click **GitHub Apps**.
5. Next to the GitHub App that you want to delete a private key for, click **Edit**.
6. Under "Private keys", to the right of the private key you want to delete, click **Delete**.
7. When prompted, confirm you want to delete the private key by clicking **Delete**. If your GitHub App has only one key, you will need to generate a new key before deleting the old key. For more information, see "Generating private keys."

Storing private keys

The private key is the single most valuable secret for a GitHub App. Consider storing the key in a key vault, such as Azure Key Vault, and making it sign-only. This helps ensure that you can't lose the private key. Once the private key is uploaded to the key vault, it can never be read from there. It can only be used to sign things, and access to the private key is determined by your infrastructure rules.

Alternatively, you can store the key as an environment variable. This is not as strong as storing the key in a key vault. If an attacker gains access to the environment, they can read the private key and gain persistent authentication as the GitHub App.

You should not hard-code your private key in your app, even if your code is stored in a private repository.

For more information, see "Best practices for creating a GitHub App."