About SSH certificate authorities

With an SSH certificate authority, your organization or enterprise account can provide SSH certificates that members and outside collaborators can use to access your resources with Git.

In this article

About SSH certificate authorities

An SSH certificate is a mechanism for one SSH key to sign another SSH key. If you use an SSH certificate authority (CA) to provide your organization members and outside collaborators with signed SSH certificates, you can add the CA to your enterprise account or organization to allow these organization contributors to use their certificates to access organization resources.

> **Note:** To use SSH certificate authorities, your organization must use GitHub Enterprise Cloud. For more information about how you can try GitHub Enterprise Cloud for free, see "Setting up a trial of GitHub Enterprise Cloud."

After you add an SSH CA to your organization or enterprise account, you can use the CA to sign client SSH certificates for organization members and outside collaborators. These organization contributors can use the signed certificates to access that organization's repositories.

Certificates added to your enterprise grant access to all organizations owned by your enterprise account. For more information, see "Enforcing policies for security settings in your enterprise."

You can require that members use SSH certificates to access organization resources, unless SSH is disabled in your repository.

Optionally, you can require that members and outside collaborators use SSH certificates to access organization resources. For more information, see "Managing your organization's SSH certificate authorities" and "Enforcing policies for security settings in your enterprise."

For example, you can build an internal system that issues a new certificate to your developers every morning. Each developer can use their daily certificate to work on your organization's repositories on GitHub Enterprise Cloud. At the end of the day, the certificate can automatically expire, protecting your repositories if the certificate is later compromised.

Organization contributors can use their signed certificates for authentication even if you've enforced SAML single sign-on (SSO), without the need to authorize the signed certificates.

Unless you make SSH certificates a requirement, organization members and outside collaborators can continue to use other means of authentication to access your organization's resources with Git, including their username and password, personal access tokens, and their own SSH keys.

Members cannot use the certificate to access forks of the organization's repositories, unless the enterprise has allowed SSH CAs to access user owned repositories. For more information, see "About SSH certificate authorities."

About SSH URLs with SSH certificates

If your organization requires SSH certificates, to prevent authentication errors, organization members and outside collaborators should use a special URL that includes the organization ID when performing Git operations over SSH. This special URL allows the client and server to more easily negotiate which key on the member's computer should be used for authentication. If a member uses the normal URL, which starts with `git@github.com`, the SSH client might offer the wrong key, causing the operation to fail.

Anyone with read access to the repository can find this URL by selecting the **Code** dropdown menu on the main page of the repository, then clicking **Use SSH**.

If your organization doesn't require SSH certificates, contributors can continue to use their own SSH keys, or other means of authentication. In that case, either the special URL or the normal URL, which starts with `git@github.com`, will work.

Issuing certificates

When you issue each certificate, you must include an extension that specifies which GitHub Enterprise Cloud user the certificate is for. You can reference the user using their login handle or their user ID. For example, you can use OpenSSH's `ssh-keygen` command, replacing KEY-IDENTITY with your key identity and USERNAME with a GitHub Enterprise Cloud username or user ID. The certificate you generate will be authorized to act on behalf of that user for any of your organization's resources. Make sure you validate the user's identity before you issue the certificate.

> **Note:** You must update to OpenSSH 7.6 or later to use these commands.

To use the `login` to identify the user, use `extension:login`:

```
ssh-keygen -s ./ca-key -V '+1d' -I KEY-IDENTITY -O
extension:login@github.com=USERNAME ./user-key.pub
```

To use the user ID, use `extension:id`:

```
ssh-keygen -s ./ca-key -V '+1d' -I KEY-IDENTITY -O extension:id@github.com=ID
./user-key.pub
```

> **Warning**: After a certificate has been signed and issued, the certificate cannot be revoked.

For CAs uploaded after March 27th, 2024, you must use the `-V` flag to configure a lifetime less than 366 days for the certificate. For CAs uploaded before this date, the `-V` flag is optional, and you can create certificates that are irrevocable and live forever.

If you have legacy CAs that are exempt from the expiration requirement, you can upgrade the CA to enforce the requirement. To learn more, see "Managing your organization's SSH certificate authorities" and "Enforcing policies for security settings in your enterprise."

If you use a username as the login extension, GitHub validates that the named user has not been renamed since the certificate was issued. This prevents a rename attack, where a certificate issued for a username is valid even if the underlying user account changes. To enforce this, the certificate must include the `valid_after` claim, which tells us when the certificate was issued. This field is often missing if an expiration is not required for the certificate, which is why expirations are now required.

To issue a certificate for someone who uses SSH to access multiple GitHub products, you can include two login extensions to specify the username for each product. For example, the following command would issue a certificate for USERNAME-1 for the user's account for GitHub Enterprise Cloud, and USERNAME-2 for the user's account on GitHub Enterprise Server at HOSTNAME.

```
ssh-keygen -s ./ca-key -V '+1d' -I KEY-IDENTITY -O
extension:login@github.com=USERNAME-1 extension:login@HOSTNAME=USERNAME-2
./user-key.pub
```

You can restrict the IP addresses from which an organization member can access your organization's resources by using a `source-address` extension. The extension accepts a specific IP address or a range of IP addresses using CIDR notation. You can specify multiple addresses or ranges by separating the values with commas. For more information, see "Classless Inter-Domain Routing" on Wikipedia.

```
ssh-keygen -s ./ca-key -V '+1d' -I KEY-IDENTITY -O
extension:login@github.com=USERNAME -O source-address=COMMA-SEPARATED-LIST-OF-
IP-ADDRESSES-OR-RANGES ./user-key.pub
```

Help and support