# Best practices for securing accounts

**In this article**



Guidance on how to protect accounts with access to your software supply chain.

## About this guide

This guide describes the highest impact changes you can make to increase account security. Each section outlines a change you can make to your processes to improve the security. The highest impact changes are listed first.

## What's the risk?

Account security is fundamental to the security of your supply chain. If an attacker can take over your account on GitHub Enterprise Server, they can then make malicious changes to your code or build process. So your first goal should be to make it difficult for someone to take over your account and the accounts of other users of your GitHub Enterprise Server instance.

## Centralize authentication

If you're the site administrator for your GitHub Enterprise Server instance, you can simplify the login experience for users by choosing an authentication method that connects with your existing identity provider (IdP), like CAS, SAML, or LDAP. This means that they no longer need to remember an extra password for GitHub.

Some authentication methods also support communicating additional information to GitHub Enterprise Server, for example, what groups the user is a member of, or synchronizing cryptographic keys for the user. This is a great way to simplify your administration as your organization grows.

For more information about the authentication methods available for GitHub Enterprise Server, see "About authentication for your enterprise."

## Configure two-factor authentication

The best way to improve the security of your personal account or your GitHub Enterprise Server instance is to configure two-factor authentication (2FA). Passwords by themselves can be compromised by being guessable, by being reused on another site that's been

compromised, or by social engineering, like phishing. 2FA makes it much more difficult for your accounts to be compromised, even if an attacker has your password.

As a best practice, to ensure both security and reliable access to your account, you should always have at least two second-factor credentials registered on your account. Extra credentials ensures that even if you lose access to one credential, you won't be locked out of your account.

If you're the site administrator for your GitHub Enterprise Server instance, you may be able to configure 2FA for all users of your instance. The availability of 2FA on GitHub Enterprise Server depends on the authentication method that you use. For more information, see "Centralize user authentication."

If you're an organization owner, then you may be able to require that all members of the organization enable 2FA.

To learn more about enabling 2FA on your own account, see "Configuring two-factor authentication." To learn more about requiring 2FA in your organization, see "Requiring two-factor authentication in your organization."

## Configure your enterprise account

Enterprise owners may be able to require 2FA for all users on the instance. The availability of 2FA policies on GitHub Enterprise Server depends on how users authenticate to access your instance.

- If you sign into your GitHub Enterprise Server instance through an external IdP using CAS or SAML SSO, you cannot configure 2FA on GitHub Enterprise Server. Someone with administrative access to your IdP must configure 2FA for the IdP.
- If you sign into your GitHub Enterprise Server instance through an external LDAP directory, you can require 2FA for your enterprise on GitHub Enterprise Server. If you allow built-in authentication for users outside of your directory, individual users can enable 2FA, but you cannot require 2FA for your enterprise.

For more information, see "Enforcing policies for security settings in your enterprise."

## Configure your personal account

> **Note**: Depending on the authentication method that a site administrator has configured for your GitHub Enterprise Server instance, you may not be able to enable 2FA for your personal account.

GitHub Enterprise Server supports several options for 2FA, and while any of them is better than nothing, the most secure option is a WebAuthn credential. WebAuthn requires an authenticator such as a FIDO2 hardware security key, a platform authenticator like Windows Hello, an Apple or Google phone, or a password manager. It's possible, although difficult, to phish other forms of 2FA (for example, someone asking you to read them your 6 digit one-time password). However WebAuthn is much more resistant to phishing, because domain scoping is built into the protocol, which prevents credentials from a website impersonating the login page from being used on GitHub Enterprise Server.

When you set up 2FA, you should always download the recovery codes and set up more than one 2FA credential. This ensures that access to your account doesn't depend on a single device. For more information, see "Configuring two-factor authentication" and "Configuring two-factor authentication recovery methods."

## Configure your organization account

**Note**: Depending on the authentication method that a site administrator has configured for your GitHub Enterprise Server instance, you may not be able to require 2FA for your organization.

If you're an organization owner, you can see which users don't have 2FA enabled, help them get set up, and then require 2FA for your organization. To guide you through that process, see:

1. "Viewing whether users in your organization have 2FA enabled"
2. "Preparing to require two-factor authentication in your organization"
3. "Requiring two-factor authentication in your organization"

## Connect to GitHub Enterprise Server using SSH keys

There are other ways to interact with GitHub Enterprise Server beyond signing into the website. Many people authorize the code they push to GitHub with an SSH private key. For more information, see "About SSH."

Just like your account password, if an attacker were able to get your SSH private key, they could impersonate you and push malicious code to any repository you have write access for. If you store your SSH private key on a disk drive, it's a good idea to protect it with a passphrase. For more information, see "Working with SSH key passphrases."

Another option is to generate SSH keys on a hardware security key. You could use the same key you're using for 2FA. Hardware security keys are very difficult to compromise remotely, because the private SSH key remains on the hardware, and is not directly accessible from software. For more information, see "Generating a new SSH key and adding it to the ssh-agent."

Hardware-backed SSH keys are quite secure, but the hardware requirement might not work for some organizations. An alternative approach is to use SSH keys that are only valid for a short period of time, so even if the private key is compromised it can't be exploited for very long. This is the concept behind running your own SSH certificate authority. While this approach gives you a lot of control over how users authenticate, it also comes with the responsibility of maintaining an SSH certificate authority yourself. For more information, see "About SSH certificate authorities."

## Next steps

- "Securing your end-to-end supply chain"
- "Best practices for securing code in your supply chain"
- "Best practices for securing your build system"
- "Best practices for preventing data leaks in your organization"