Table of Contents

[Authenticate and authorize user identities on GitHub 2](#_Toc132915517)

[Learning objectives 2](#_Toc132915518)

[Introduction 3](#_Toc132915519)

[Learning goals 3](#_Toc132915520)

[Prerequisites 3](#_Toc132915521)

[User identity and access management 4](#_Toc132915522)

[Organization management through SAML SSO 4](#_Toc132915523)

[Organization private information 5](#_Toc132915524)

[User authentication 6](#_Toc132915525)

[Authentication with SAML SSO 6](#_Toc132915526)

[Enforce SAML SSO for your organization 6](#_Toc132915527)

[Multifactor authentication or 2FA 7](#_Toc132915528)

[Options to identify with 2FA 8](#_Toc132915529)

[Audit 2FA for user compliance 9](#_Toc132915530)

[User authorization 11](#_Toc132915531)

[Authorization with SAML SSO through SCIM 11](#_Toc132915532)

[SSH key and PAT with SAML SSO 12](#_Toc132915533)

[Connect your IdP to your organization 13](#_Toc132915534)

[Team synchronization 14](#_Toc132915535)

[Usage limits 15](#_Toc132915536)

[Enable team synchronization 15](#_Toc132915537)

[Disable team synchronization 16](#_Toc132915538)

[Summary 17](#_Toc132915539)

[Learn more 17](#_Toc132915540)

**Authenticate and authorize user identities on GitHub**

This module provides an overview of the authentication and authorization options available to you in your GitHub organization or GitHub Enterprise.

## Learning objectives

By the end of this module, you will:

* Be able to describe the Authentication and Authorization Model
* Understand how to manage user access to your GitHub organization through Authorization and Authentication tools
* Identify the supported identity providers and technologies that support secure repository access
* Understand the implications of enabling SAML SSO
* Identify the authorization and authentication options available, and understand the administrator’s role in enforcement of a secure access strategy for a GitHub enterprise
* Describe how users access private information in a GitHub organization
* Evaluate the benefits of enabling Team Synchronization to manage team membership

# Introduction

User authentication has traditionally been achieved using a User ID and password. A password is a single-factor form of authentication. The fundamental issue with single-factor authentication is that it's easier for any bad actor with knowledge of the sign-on information to impersonate the valid user. To prevent a breach of security for a user account, there are authentication tools available on GitHub to promote security best practices. You can even enforce a security policy for all GitHub users within the organization.

Controlling access to your company's data is foundational for a secure GitHub Enterprise. GitHub is committed to helping enterprises on their security journey with authentication methods to allow for more secure account access and a better user experience. In a GitHub Enterprise, most organizations want to require extra levels of authentication for better security. Enterprise System Administrators can enforce authentication and authorization security policies across an organization. These security features allow you to ensure that users are required to sign on securely to access the correct permissions in repositories. These features also include access and tools for auditing user access and activity, identity maintenance, and authentication compliance. As an administrator, you should work with your internal resources to identify what type of authentication and authorization is appropriate. This module provides an overview of the authentication and authorization options available to you in your GitHub organization or GitHub Enterprise.

## Learning goals

By the end of this module, you'll be able to:

* Describe the GitHub authentication and authorization Model.
* Understand how to manage user access to your GitHub organization through Authorization and Authentication tools.
* Identify the identity providers and technologies that support secure repository access.
* Understand the implications of enabling SAML SSO.
* Identify the authorization and authentication options available, and understand the administrator’s role in enforcing a secure access strategy for a GitHub enterprise.
* Describe how users access private information in a GitHub organization.
* Evaluate the benefits of enabling Team Synchronization to manage team membership.

## Prerequisites

* Administrative access to your GitHub organization or GitHub Enterprise

# User identity and access management

Controlling access to your company's data and applications is the foundation of your enterprise security strategy. Every GitHub Enterprise user's experience begins with logging in. Before a user can access your GitHub organization, they must authenticate by supplying credentials that confirm their identity. For an individual account, users can log in with only a user name and password, but every GitHub user can and should enable 2FA (two-factor authentication) for a more secure authentication process.

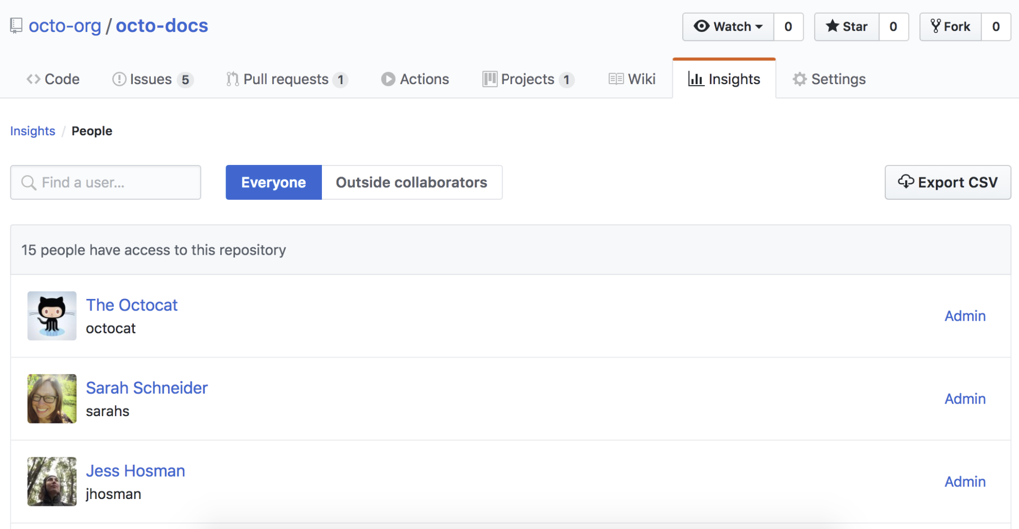
A challenge you may face in your organization is how to balance the ease of use with an authorization process while maintaining secure best practices. Setting up your team for success requires ease of access for the user under the umbrella of secure organizational requirements.

Configuring authentication is the first step in ensuring secure software development in your enterprise. The good news is that using the tools available with your IdP is critical to getting the most value from GitHub.

## Organization management through SAML SSO

An important component of an enterprise security strategy is SAML SSO. It provides a link between the IdP authorization and access to service providers (SaaS). This form of authentication allows users to sign in to all their applications with one set of credentials. Through SAML, the IdP authenticates users and grants authorization to services like GitHub. When a user logs into GitHub, they can view the enterprises of which they're members. However, if the user tries to access repository data, it will prompt for enterprise credentials (Enterprise ID).

As the Enterprise Administrator, you're responsible for the authorization of user access and permissions. Limiting a user's access to only the resources they need is important when securing your repository. This responsibility can also include routine audit events and maintaining tightly scoped access. As an administrator of a repository, you have an overview of every user with their specific access within the repository. You can also easily export this data to a CSV file.



You need to configure SAML SSO for a GitHub organization with the IdP you're using. If you have specific questions on how to implement SAML SSO with your chosen IdP, you can find details in the documentation for each supported IdP. Below is a list of the SAML IdPs that GitHub currently supports:

* Active Directory Federation Services (AD FS)
* Azure Active Directory (Azure AD)
* Okta
* OneLogin
* PingOne
* Shibboleth

Note: GitHub offers limited support for all identity providers that implement the SAML 2.0 standard.

You can accomplish more access management when using multiple organizations. You can use organizations to create distinct groups of users within your company, such as divisions or groups working on similar projects. Public and internal repositories that belong to an organization are accessible to members of other organizations in the enterprise. Private repositories are inaccessible to anyone who isn't a member of the organization.

## Organization private information

When an organization member creates a repository, they can choose to make the repository public or private. In addition, when creating the repository in an organization that's owned by an enterprise account, they can choose to make the repository internal. Public repositories are accessible to everyone on the internet. Private repositories are only accessible to the user who created the repository and the people with whom they explicitly share access. Keep in mind that certain organization members have access to organization internal repositories.

# User authentication

Here, you'll learn about two recommended authentication systems employed by GitHub Enterprises. You'll also learn how to do a basic but important 2FA audit for user compliance.

When it comes to user authentication, security should be the number one consideration that comes to mind. Strong security is essential. It seems like every month or so, a company reports a data breach. Credentials are stolen because of inefficient security processes, or simply because of a lack of up-to-date security features within the company. Establishing secure user authentication can be a difficult task if user adoption requires long and frustrating steps to authenticate. Ask any security professional and they'll mention several ways that companies authenticate their users.

There are two recommended authentication methods you can implement when authenticating your users on GitHub: SAML SSO and multifactor authentication, also known as 2FA.

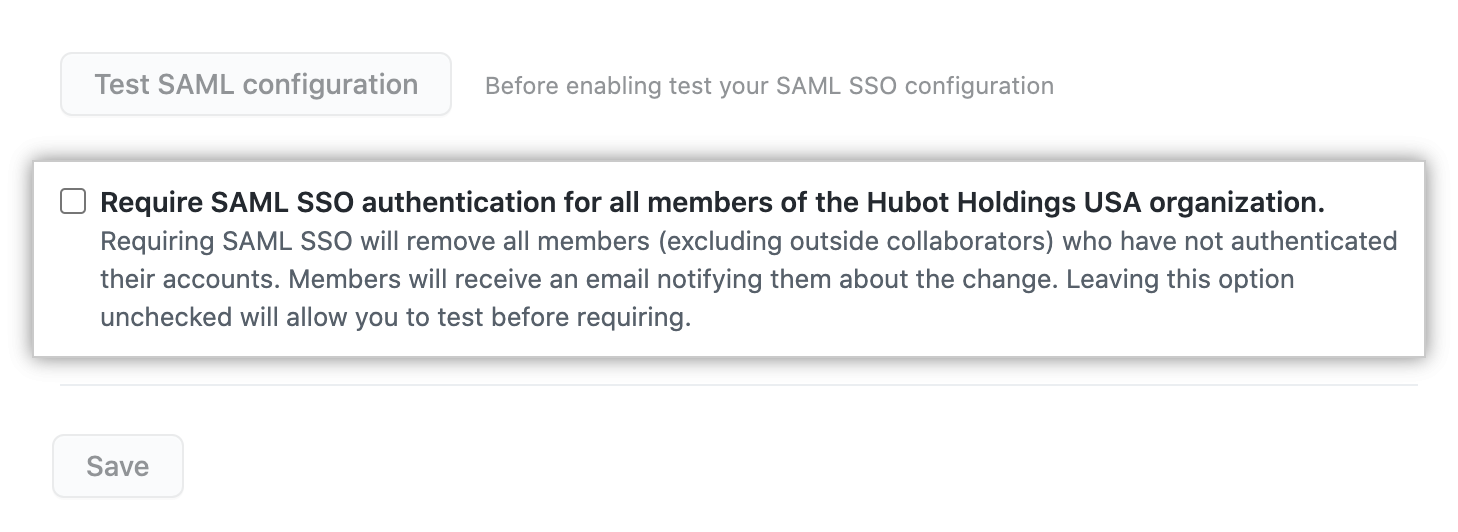
## Authentication with SAML SSO

SAML authentication is a process used to verify user identity and credentials against a known identity provider. Nowadays, your company may likely already be using this form of authentication. If so, you can link your existing IdP to GitHub for user sign-in management. The following is a high-level overview of the process that occurs with enabled SAML SSO within your GitHub Enterprise:

* Before enabling SAML SSO with your GitHub Enterprise, an Administrator needs to connect the GitHub organization to a supported IdP.
* Next, when a member accesses resources within an organization that uses SAML SSO, GitHub redirects the member to the IdP to authenticate.
* After successful authentication, the IdP redirects the member back to GitHub, where the member can access the organization's resources. The result means that even after configuring SAML SSO, members of the GitHub organization will continue to be prompted to log into their user accounts on GitHub.

### Enforce SAML SSO for your organization

It's important to note that if you have enabled SAML SSO across your organization, you'll need to enforce authentication after the setting has been enabled. This enforcement will make the authentication process a requirement. As the organization administrator, you can enforce this setting by selecting Your organizations, then selecting Settings, and then choosing Organization security. Under SAML single sign-on, select Require SAML SSO authentication for all members of the organization.



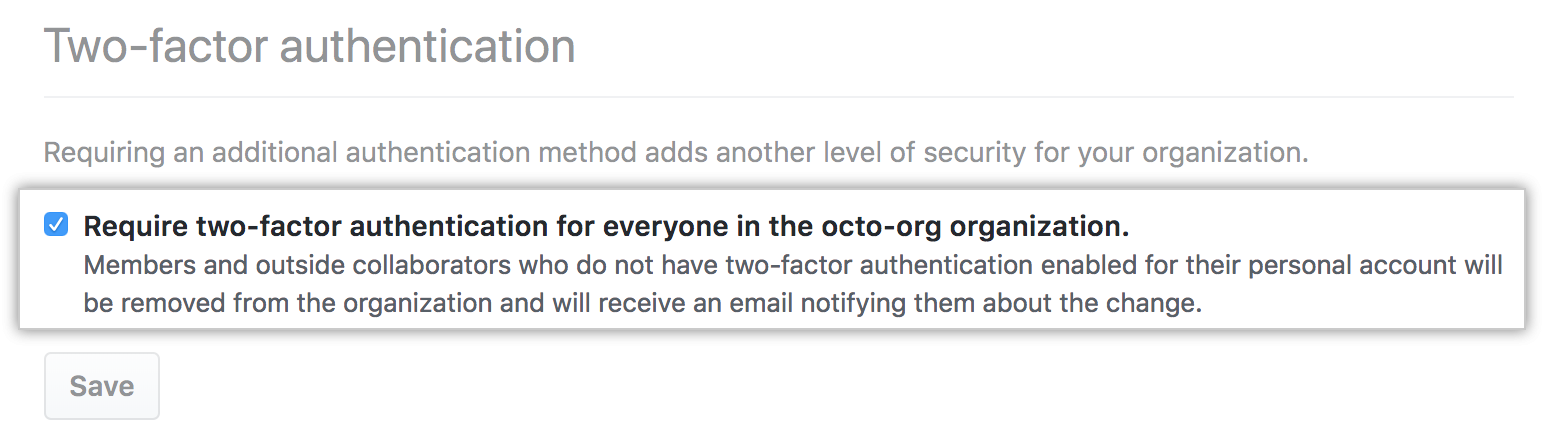
After you enforce authentication for an organization, GitHub will then remove any members of the organization that haven't authenticated successfully with your SAML IdP.

Important

Keep in mind that GitHub will not remove members of the enterprise that have not yet authenticated successfully with your SAML IdP. The member will be prompted to authenticate with your SAML IdP the next time the member accesses the enterprise's resources or tries to authenticate after the setting has been enabled.

## Multifactor authentication or 2FA

Two-factor authentication is an extra level of security available to GitHub Enterprise accounts. With 2FA, a member in your organization is required to log in with username and password, and also provide a secondary form of authentication. This second form of authentication needs to be something only the user knows or has access to. You can require organization members, outside collaborators, and billing managers to enable two-factor authentication for their personal accounts by selecting an organization security setting.



Keep in mind that before you can require organization members, outside collaborators, or billing managers to use two-factor authentication, they must enable the two-factor authentication setting for their account on GitHub. You'll need to communicate this requirement to your team via email or some other means of communication.

Warning: When you require the use of two-factor authentication for your organization, all accounts that do not use 2FA will be removed from the organization and lose access to its repositories. This includes bot accounts.

For more detailed information about 2FA, see [Securing your account with two-factor authentication (2FA)](https://docs.github.com/en/github/authenticating-to-github/securing-your-account-with-two-factor-authentication-2fa).

### Options to identify with 2FA

Only users who legitimately need access to your organization's data should have it. Insufficient data security could lead to an attack or breach of important data. One way to ensure that your organization is meeting basic security requirements is by enabling 2FA. When you require 2FA, a malicious actor needs more than just a user's password and username to access an account. 2FA makes it much harder for malicious actors to access your organization's data, because they don't have access to your users' physical devices.

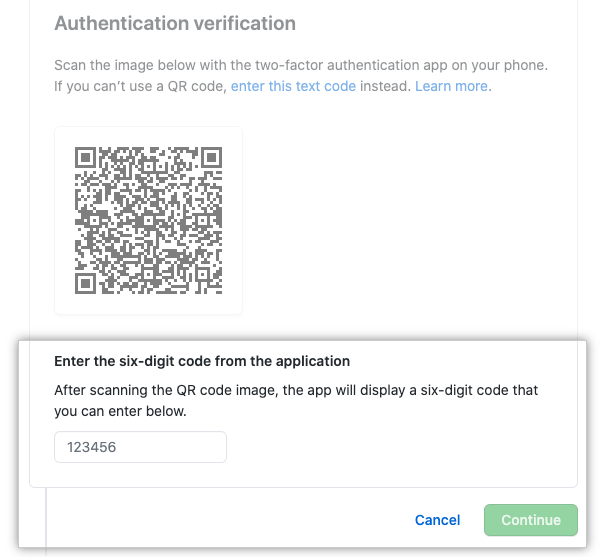
The codes used for 2FA expire in a short amount of time. Once expired, the login attempt is no longer valid, and the access request is canceled. By enabling 2FA, your user's accounts will be better protected from malicious attacks and data breaches. In your GitHub enterprise, your users have three ways that they can authenticate with 2FA: security keys, TOTP, and SMS.

#### Security keys

With security keys, your enterprise can achieve a higher level of user security and protection. With two-factor authentication enabled, security keys provide a strong, convenient, and phishing-proof option for 2FA. Authentication with a security key requires that TOTP or SMS authentication has already been completed. On most devices, you can use a physical security key over USB or NFC. A user can register a new security key by accessing their profile setting and following the security key’s documentation. Using these keys is the most secure form of 2FA, because they're nearly impossible for a malicious party to replicate. When using a security key, none of the sensitive information ever leaves the physical security key device. Authentication with a security key is the most secure way to authenticate, but your users will need to complete one of the following authentication methods to configure the key.

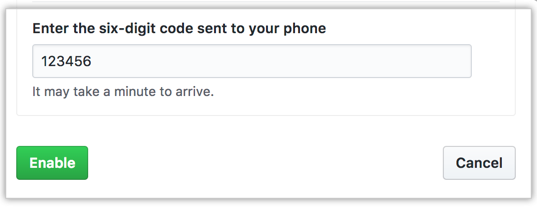
#### TOTP

GitHub recommends using a cloud-based TOTP app to configure 2FA. TOTP applications are more reliable than SMS, especially for locations outside the United States. TOTP apps support the secure backup of your authentication codes in the cloud, and can be restored if you lose access to your device.



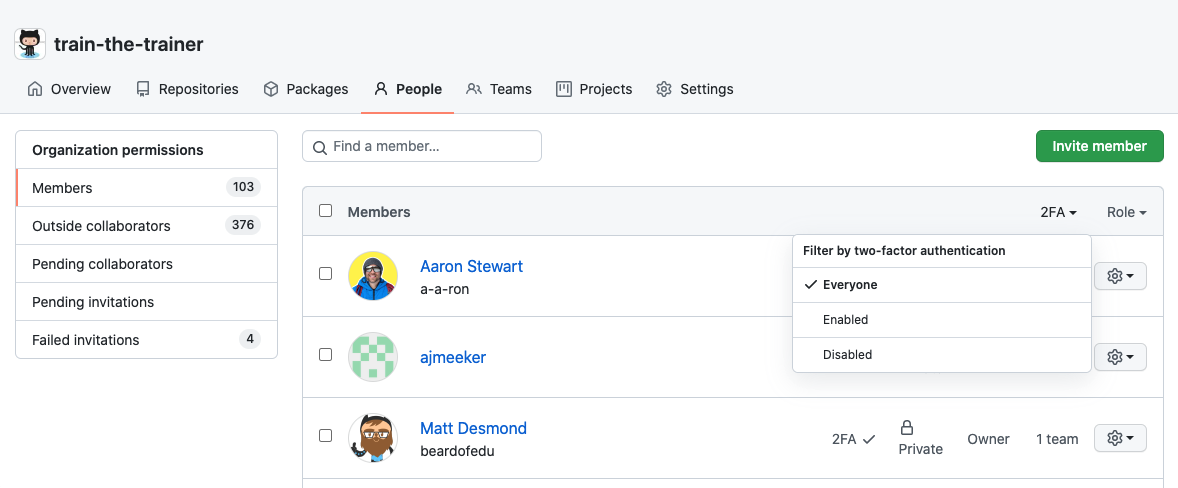
#### SMS

If your users aren't able to authenticate using a TOTP mobile app, they can authenticate using SMS messages. This form of 2FA relies on the assumption that the user is the only person with access to their mobile device. GitHub doesn't support authentication via SMS in all countries/regions. Before an Administrator allows users to authenticate via SMS, they should confirm that it's supported in the country/region where the users are located.



### Audit 2FA for user compliance

You can review which organization owners, members, and outside collaborators have enabled two-factor authentication by navigating to the right corner of GitHub.com, clicking your profile photo, clicking **Your organizations**, then selecting the name of the chosen organization. Under the organization name, click the **People** tab and then select the **2FA** option. From here, you can see which members in the organization have enabled two-factor authentication, and which outside contributors have it enabled.



Revoking access for users in your enterprise who aren't compliant is possible, but you'll need to contact them outside of GitHub to relay the reason they no longer have access to your organization. Communication to non-compliant users is traditionally done through email notification.

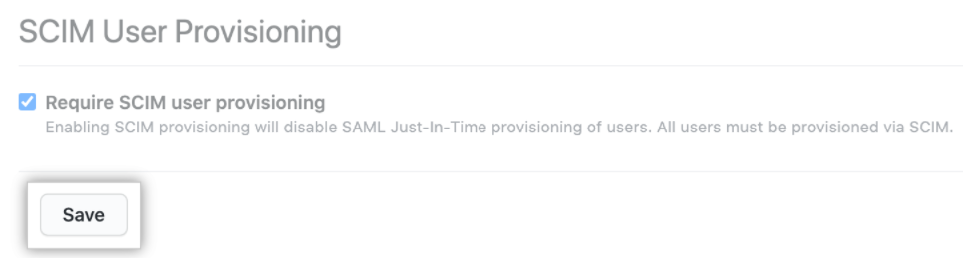
# User authorization

Here, you'll be introduced to the authorization for GitHub users, which is the next step after user authentication.

As we learned previously, a user can be granted access to a GitHub organization that uses SAML single sign-on (SSO) by authenticating through the identity provider (IdP). But after you authenticate the user with the IdP successfully from GitHub, you must now authorize any personal access token, SSH key, or OAuth App that you would like the user to use to access your organization's resources. Let's look a little deeper into this authentication process with SAML single sign-on.

## Authorization with SAML SSO through SCIM

SAML single sign-on (SSO) gives organization owners and enterprise owners on GitHub a way to control and secure access to organization resources like repositories, issues, and pull requests. If you use SAML SSO in your organization, you can implement SCIM or System for Cross-domain Identity Management. SCIM lets you add, manage, or remove organization member access within GitHub. SCIM was developed to allow synchronization of information between an IdP and multiple applications.



SCIM is a protocol that tells the directory an account has been created and allows you to automate the exchange of user identity information between systems. For example, while onboarding a new employee, using a central directory allows you to automatically provision the user to access services like GitHub. An administrator can deprovision an organization member using SCIM and automatically remove the member from the organization.

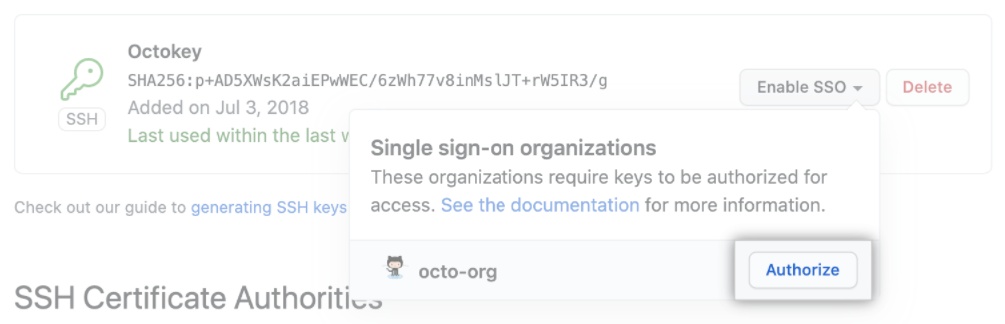
Note: If you use SAML SSO without implementing SCIM, you won't have automatic deprovisioning.

SCIM integrations allow the secure exchange of user identity data between your IdP and your enterprise on GitHub. SCIM was developed to allow the synchronization of information between an IdP and multiple applications. When organization members' sessions expire after their access is removed from the IdP, they aren't automatically removed from the organization. Authorized tokens grant access to the organization even after their sessions expire. To remove this access, you can either manually remove the authorized token from the organization or automate its removal with SCIM.

### SSH key and PAT with SAML SSO

Once these systems are implemented, user lifecycle-management changes made in your IdP will be reflected in your GitHub enterprise automatically. The SAML IdP and the SCIM client must use matching NameID and userName values for each user. This process allows a user authenticating through SAML to be linked to their provisioned SCIM identity. Each time you make changes to group membership in your IdP, the IdP will make a SCIM call to GitHub.com to update the corresponding organization's membership. Each time an enterprise member accesses a resource protected by your enterprise account's SAML configuration, that SAML assertion will trigger provisioning.

To access your organization's protected resources using the API and Git on the command line, your users will need to authorize and authenticate with a PAT (personal access token) or SSH key. Users can authorize an existing PAT or SSH key, or create a new PAT or SSH key and then authorize it. As the administrator, you're able to review each personal access token and SSH key that a member has authorized for API and Git access.



After you enable SAML SSO, there are multiple ways you can add new members to your organization. Organization owners can invite new members manually on GitHub or by using the API. To provision new users without an invitation from an organization owner, you can use the URL https://github.com/orgs/ORGANIZATION/sso/sign\_up, replacing ORGANIZATION with the name of your organization. For example, you can configure your IdP so that anyone with access to the IdP can click a link on the IdP's dashboard to join your GitHub organization.

SCIM and SAML SSO also have security benefits. The first time a member uses SAML SSO to access your organization, GitHub automatically creates a record that links your organization, the member's GitHub account, and the member's account on your IdP. You can review the SAML SSO identity that a member has linked to their GitHub account. When available, the entry will include SCIM data for the user. You can also view and revoke the linked SAML identity, active sessions, and authorized credentials for members of your organization or enterprise account. When an employee leaves the company, the off-boarding process is automated and deprovisioning happens automatically. When you unassign the application from a user or deactivate a user's account on your IdP, it will communicate with GitHub.com to invalidate any SAML sessions for that user. The automation of these tasks reduces the time required for an administrator to manage user credentials, and the risk of error associated with manually entering and updating user information is also mitigated.

### Connect your IdP to your organization

To use SAML single sign-on and SCIM, you must connect your identity provider to your GitHub organization. When you enable SAML SSO for your GitHub organization, you connect your identity provider to your organization. Keep in mind that not all SAML identity providers are currently supported by GitHub for SCIM. Following is a list of the GitHub supported identity providers for SCIM:

* Azure AD
* Okta
* OneLogin

If your support request is outside of the scope of what our team can help you with, we may recommend next steps to resolve your issue outside of GitHub Support. Your support request is possibly out of GitHub Support's scope if the request is primarily about:

* Third-party integrations
* Hardware setup
* CI/CD, such as Jenkins
* Writing scripts
* Configuration of external authentication systems, such as SAML identity providers
* Open-source projects

Some companies may have built their own system and synchronize their accounts in a custom way. GitHub doesn't support these custom-created accounts with technical assistance. Technical support by GitHub for changes in the way GitHub.com uses SCIM and SAML is available for enterprises that are using one of the providers listed previously.

If you're uncertain if the issue is out of scope, open a ticket with GitHub support, and they'll be happy to help you determine the best way to continue.

# Team synchronization

If your company is using Azure AD or Okta as your IdP for your enterprise in GitHub's cloud, you can use team synchronization to manage team membership within each organization through IdP groups. With team synchronization enabled, changes made in an IdP group are automatically reflected on GitHub. This feature reduces the need for manual updates and custom scripts. You can centrally manage users' identities, allowing authorization, review, and revocation of permissions.

When you synchronize a GitHub team with an IdP group, changes to the IdP group are reflected on GitHub automatically, reducing the need for manual updates and custom scripts. You can use an IdP with team synchronization to manage administrative tasks such as onboarding new members, granting new permissions for movements within an organization, and removing member access to the organization.

Managing a team via your service provider allows you to save time and resources that you'd otherwise spend duplicating in GitHub the information about your team that's already captured in your IdP. The Administrator of your IdP will need to enable SAML SSO and SCIM to implement team synchronization.

| **Features** | **Description** |
| --- | --- |
| Sync Users | Add or remove users from Teams in GitHub to keep in sync with Active Directory groups |
| Sync on new team | Synchronize users when a new team is created |
| Custom team/group maps | The team slug and group name will be matched automatically, unless you define a custom mapping with syncmap.yml |
| Dynamic Config | Utilize a settings file to derive Active Directory and GitHub settings |

Note

Team synchronization is available for organizations and enterprise accounts using GitHub Enterprise Cloud.

## Usage limits

When using the team synchronization feature, there are specific usage limits you need to know about. Exceeding these limits can lead to unexpected performance, and may cause synchronization failures.

* Maximum number of members in a GitHub team: 5,000
* Maximum number of members in a GitHub organization: 10,000
* Maximum number of teams in a GitHub organization: 1,500

## Enable team synchronization

With team synchronization, you can use your IdP to manage administrative tasks like onboarding new members, granting new permissions in your organization, and removing member access. When you synchronize a GitHub team with an IdP group, changes made to the IdP group are reflected on GitHub automatically, reducing the need for manual updates and custom scripts. The steps to enable team synchronization depend on the IdP you use.

You can enable and use team synchronization, but only with the following supported IdPs:

* Azure AD
* Okta

The steps to enable team synchronization depend on the IdP you want to use. There are prerequisites to enable team synchronization that apply to each IdP. To enable team synchronization with your IdP, you must obtain administrative access or work with your IdP administrator to configure the IdP integration and groups. After you enable team synchronization, team maintainers and organization owners can connect a team to an IdP group on GitHub or through the API.

**Azure AD**: The GitHub System Admin for the GitHub organization will need to identify and work with the Azure AD Administrator to configure Team Synchronization. On the Azure AD side, the service is called “automatic user account provisioning.” To enable team synchronization for Azure AD, the installation needs the following permissions:

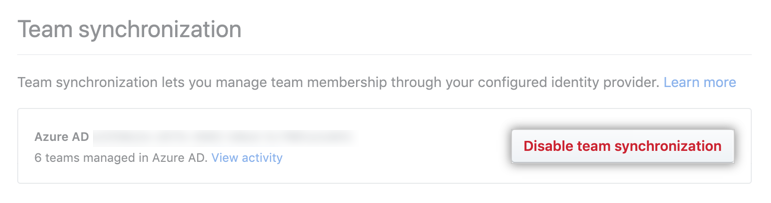
* Read all users’ full profiles
* Sign in and read user profiles
* Read directory data

**Okta**: To enable team synchronization for Okta, you or your IdP administrator must:

* Enable SAML SSO and SCIM for your organization using Okta.
* Provide the tenant URL for your Okta instance.
* Generate a valid SSWS token with read-only admin permissions for your Okta installation as a service user.

## Disable team synchronization

When you disable team synchronization, any team members who were assigned to a GitHub team through the IdP group are removed from the team and may lose access to your organization's repositories. You can disable this feature through the organization settings by selecting **Your organization** and selecting **Settings**. Next, select **Organization security** and choose **Disable team synchronization**.



# Summary

In this module, you learned about allowing access to your users, and how the authentication systems available for your GitHub organization help keep your sensitive data secure. You also learned about auditing which users and teams have access to the repositories in your organization. The goal for you as a GitHub administrator should be to give your users access to your enterprise data with robust security restrictions that are painless to use. Securing who has access to your organization will ensure that only the users who legitimately need access to your organization's data will have it.

You learned:

* How SAML SSO and 2FA are more secure than username/password authentication
* Which identity providers are supported by GitHub
* How user authorization with SCIM is supported by GitHub
* What options users have to identify with two-factor authentication
* How team synchronization through your IdP can automate team membership and help keep access to your data secure

The goal of managing access to your enterprise is to create a strong and secure GitHub development environment for your users. Without these authorization and authentication tools in place, your enterprise could be compromised by bad actors who take advantage of the susceptibility of username and password vulnerabilities to access your data. Use the security features you learned about in this module to build a secure way to authenticate and authorize your users within your organization. These systems of authentication and authorization, along with team synchronization, will allow you to ensure organizational security, control user lifecycle management, and automate the user onboarding and off-boarding process with efficiency and security.

## Learn more

Here are some links to more detailed information on the topics we discussed in this module:

* [Managing SAML single sign-on for your organization - GitHub Docs](https://docs.github.com/en/organizations/managing-saml-single-sign-on-for-your-organization)
* [Viewing and managing a member's SAML access to your organization - GitHub Docs](https://docs.github.com/en/organizations/granting-access-to-your-organization-with-saml-single-sign-on/viewing-and-managing-a-members-saml-access-to-your-organization)
* [Preparing to require two-factor authentication in your organization - GitHub Docs](https://docs.github.com/en/organizations/keeping-your-organization-secure/preparing-to-require-two-factor-authentication-in-your-organization)
* [Authorizing a personal access token for use with SAML single sign-on - GitHub Docs](https://docs.github.com/en/github/authenticating-to-github/authenticating-with-saml-single-sign-on/authorizing-a-personal-access-token-for-use-with-saml-single-sign-on)
* [Authorizing an SSH key for use with SAML single sign-on - GitHub Docs](https://docs.github.com/en/github/authenticating-to-github/authenticating-with-saml-single-sign-on/authorizing-an-ssh-key-for-use-with-saml-single-sign-on)
* [Synchronizing a team with an identity provider - GitHub Docs](https://docs.github.com/en/organizations/organizing-members-into-teams/synchronizing-a-team-with-an-identity-provider-group)