Table of Contents

[Introduction to GitHub administration 2](#_Toc132913502)

[Learning objectives 2](#_Toc132913503)

[Introduction 3](#_Toc132913504)

[Learning objectives 3](#_Toc132913505)

[What is GitHub administration? 4](#_Toc132913506)

[Administration at team level 4](#_Toc132913507)

[Best practices for team-level administration 4](#_Toc132913508)

[Administration at organization level 5](#_Toc132913509)

[Administration at enterprise level 6](#_Toc132913510)

[Learn more 6](#_Toc132913511)

[How does GitHub authentication work? 7](#_Toc132913512)

[GitHub's authentication options 7](#_Toc132913513)

[Username and password 7](#_Toc132913514)

[Personal access tokens 7](#_Toc132913515)

[SSH keys 7](#_Toc132913516)

[Deploy keys 8](#_Toc132913517)

[GitHub's added security options 8](#_Toc132913518)

[Two-factor authentication 8](#_Toc132913519)

[SAML SSO 9](#_Toc132913520)

[LDAP 9](#_Toc132913521)

[How does GitHub organization and permissions work? 10](#_Toc132913522)

[Repository permission levels 10](#_Toc132913523)

[Team permission levels 11](#_Toc132913524)

[Organization permission levels 12](#_Toc132913525)

[Enterprise permission levels 13](#_Toc132913526)

[Summary 14](#_Toc132913527)

[Learn more 14](#_Toc132913528)

# Introduction to GitHub administration

Understand the security and control measures available to GitHub administrators within an organization or enterprise.

**Learning objectives**

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

* Summarize the organizational structures and permission levels that GitHub administrators can use to organize members in order to control access and security.
* Identify the various technologies that enable a secure authentication strategy allowing administrators to centrally manage repository access.
* Describe the technologies required to centrally manage teams and members using existing directory information services.
* Describe how you can use GitHub itself as an identity provider for authentication and authorization.

# Introduction

GitHub administrators work to protect their organization's code and content assets while providing each team access to the repositories they rely on to collaborate and share their work.

Imagine that your CIO asks you for an adoption plan to help the entire company benefit from GitHub. You want to ensure every group has adequate access to the right repositories and that there's a sustainable way to provide adequate permissions to the appropriate software development and content teams. You'll need to think through the kinds of tasks that administrators need to perform and assign them the right level of access. But first, you really need to understand what options are available to you from GitHub.

In this module, you'll learn about:

* GitHub administrative tasks and their purpose at each hierarchical level.
* The various ways that administrators can configure authentication so that users can access GitHub via the web browser and the git client.
* Hierarchical permission levels and what these permissions allow you to do in GitHub.

## Learning objectives

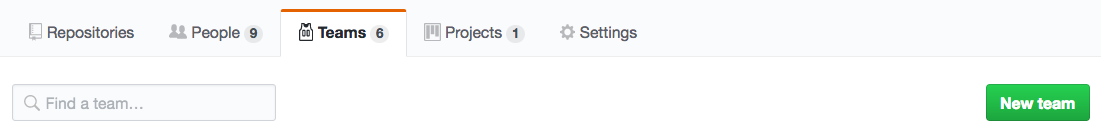
By the end of this module, you'll be able to:

* Summarize the organizational structures and permission levels that GitHub administrators can use to organize members to control access and security.
* Identify the various technologies that enable a secure authentication strategy, allowing administrators to centrally manage repository access.
* Describe the technologies required to centrally manage teams and members using existing directory information services and how you can use GitHub itself as an identity provider for authentication and authorization.

# What is GitHub administration?

As a GitHub administrator, your goal is to keep everything working smoothly for your users. In this unit, you'll learn about the different levels in the GitHub organizational hierarchy and the administration tasks associated with each level.

## Administration at team level



In GitHub, each user is an organization member that you can add to a team. You can create teams in your organization with cascading access permissions and mentions to reflect your company or group's structure. Teams are useful for refining repository permissions on a more granular level and enabling communication and notification between team members.

Additionally, GitHub allows you to sync your teams with identity provider (IdP) groups such as Azure Active Directory (Azure AD). When you synchronize a GitHub team with Azure AD, you can replicate changes to GitHub automatically, which reduces the need for manual updates and custom scripts. You can use Azure AD with team synchronization to manage administrative tasks such as onboarding new members, granting new permissions, and removing member access to the organization.

Members of a team with team maintainer or repository admin permissions can:

* Create a new team, and select or change the parent team.
* Delete or rename a team.
* Add or remove organization members from a team, or synchronize a GitHub team's membership with an IdP group.
* Add or remove outside collaborators (people who aren't explicitly members of your organization, such as consultants or temporary employees) from team repositories.
* Enable or disable team discussions on the team's page.
* Change the visibility of the team within the organization.
* Manage automatic code review assignment for pull requests, utilizing GitHub's review assignment routing algorithm.
* Schedule reminders.
* Set the team profile picture.

### Best practices for team-level administration

Creating teams in your organization enables greater flexibility for collaboration and can make it easier to separate repositories and permissions. The following are some best practices for setting up teams on GitHub:

* Create nested teams to reflect your group or company's hierarchy within your GitHub organization.
* Create teams based on interests or specific technology (JavaScript, data science, etc.) to help streamline PR review processes. Individuals can choose to join these teams according to their interests or skills.
* Enable team synchronization between your identity provider (IdP) and GitHub to allow organization owners and team maintainers to connect teams in your organization with IdP groups. When you synchronize a GitHub team with an IdP group, you can replicate changes to 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, and removing member access to the organization.

## Administration at organization level

In GitHub, organizations are shared spaces enabling users to collaborate across many projects at once. Owners and administrators can manage member access to the organization's data and repositories with sophisticated security and administrative features.

Members of an organization with the owner permission can perform a wide range of activities at the organization level including:

* Invite users to join the organization and remove members from the organization.
* Organize users into a team, and grant "team maintainer" permissions to organization members.
* Add or remove outside collaborators (people who aren't explicitly members of your organization, such as consultants or temporary employees) to organizational repositories.
* Grant repository permission levels to members, and set the base (default) permission level for a given repository.
* Set up organization security.
* Set up billing or assign a billing manager for the organization.
* Extract various types of information about repositories via the use of custom scripts.
* Apply organization-wide changes such as migrations via the use of custom scripts.

We recommend setting up only one organization for your users and repositories. If specific constraints in your company require you to create multiple organizations, be aware that:

* It isn't possible to duplicate an organization or share configurations between two organizations. This means that you must set up everything from scratch every time you create an organization, which increases the risk of errors in your settings.
* Depending on your software providers' policies, you might incur extra costs if you need to install some applications in multiple organizations.
* Managing multiple organizations is generally more difficult!

## Administration at enterprise level

Enterprise accounts include GitHub Enterprise Cloud and Enterprise Server instances and enable owners to centrally manage policy and billing for multiple organizations.

At the enterprise level, members of an enterprise with the owner permissions can:

* Enable SAML single sign-on for their enterprise account, allowing each enterprise member to link their external identity on your identity provider (IdP) to their existing GitHub account.
* Add or remove organizations from the enterprise.
* Set up billing or assign a billing manager for all organizations in the enterprise.
* Set up repository management policies, project board policies, team policies, and other security settings that apply to all the organizations, repositories, and members in the enterprise.
* Extract various types of information about organizations via the use of custom scripts.
* Apply enterprise-wide changes such as migrations via the use of custom scripts.

## Learn more

Read more about [nested teams](https://docs.github.com/organizations/organizing-members-into-teams/about-teams#nested-teams)in GitHub Docs.

# How does GitHub authentication work?

In the previous unit, you learned about typical administration tasks at the team, organization, and enterprise level. In this unit, you'll deep dive into one of the most common administrative tasks performed by organization owners: setting up and controlling users' authentication to GitHub.

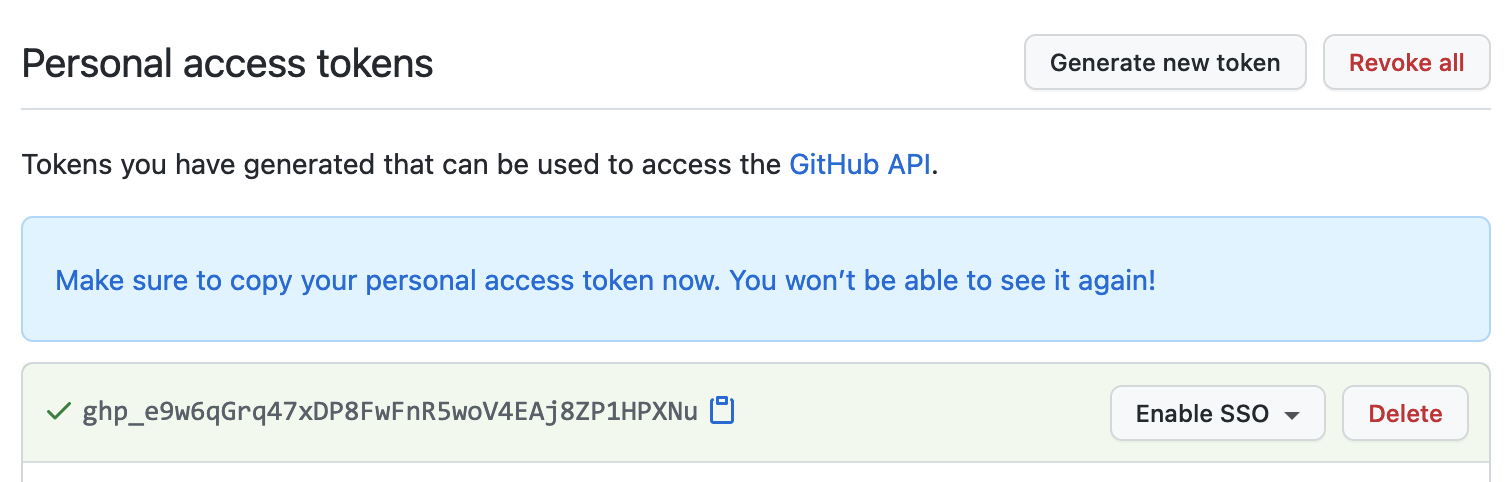
## GitHub's authentication options

There are several options for authenticating with GitHub:

### Username and password

Administrators can allow users to continue using the default username and password authentication method, sometimes known as the "basic" HTTP authentication scheme. In recent years, basic authentication has proven to be too risky when dealing with highly sensitive information, so we strongly recommend using one (or several) of the other options listed in this unit.

### Personal access tokens



Personal access tokens (PATs) are an alternative to using passwords for authentication to GitHub when using the GitHub API or the command line. Users generate a token via the GitHub's settings option, and tie the token permissions to a repository or organization. When users interact with GitHub by using the git command-line tool, they can enter the token information when they're asked for their username and password.

### SSH keys

As an alternative to using personal access tokens, users can connect and authenticate to remote servers and services via SSH with the help of SSH keys. SSH keys eliminate the need for users to supply their username and personal access token for every interaction.

When setting up SSH, users generate an SSH key, add it to the ssh-agent and then add the key to their GitHub account. Adding the SSH key to the ssh-agent ensures that the SSH key has a passphrase as an extra layer of security. Users can configure their local copy of git to automatically supply the passphrase, or they can supply it manually each time they use the git command-line tool to interact with GitHub.

You can even use SSH keys with a repository owned by an organization that uses SAML single sign-on. If the organization provides SSH certificates, users can also use it to access the organization's repositories without adding the certificate to their GitHub account.

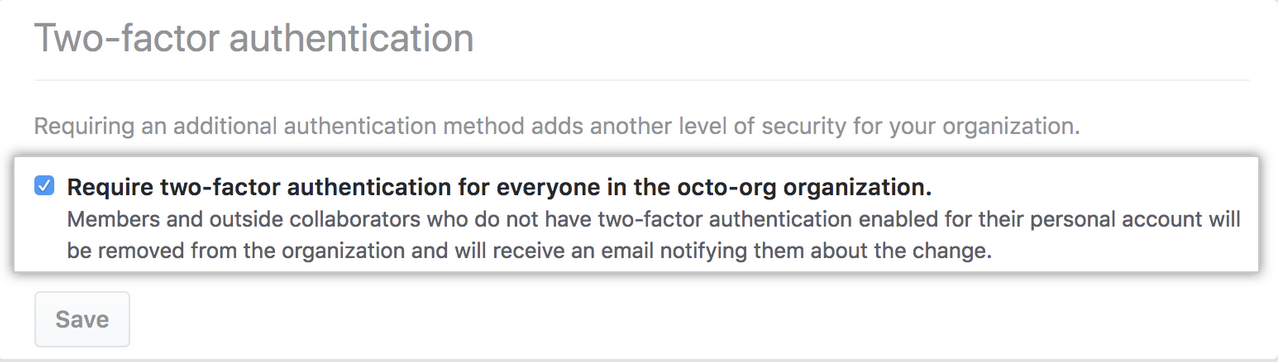
### Deploy keys

Deploy keys are another type of SSH key in GitHub that grants a user access to a single repository. GitHub attaches the public part of the key directly to the repository instead of a personal user account, and the private part of the key remains on the user's server. Deploy keys are read-only by default, but you can give them write access when adding them to a repository.

## GitHub's added security options

GitHub also offers the following extra security options.

### Two-factor authentication



Two-factor authentication (2FA), sometimes known as multifactor authentication (MFA), is an extra layer of security used when logging into websites or apps. With 2FA, users have to sign in with their username and password and provide another form of authentication that only they have access to.

For GitHub, the second form of authentication is a code generated by an application on a user's mobile device or sent as a text message (SMS). After a user enables 2FA, GitHub generates an authentication code anytime someone attempts to sign into their GitHub account. Users can only sign into their account if they know their password and have access to the authentication code on their phone.

Organization owners can require organization members, outside collaborators, and billing managers to enable two-factor authentication for their personal accounts, making it harder for malicious actors to access an organization's repositories and settings.

Enterprise owners can also enforce certain security policies for all organizations owned by an enterprise account.

### SAML SSO

If you centrally manage your users' identities and applications with an identity provider (IdP), you can configure Security Assertion Markup Language (SAML) single sign-on (SSO) to protect your organization's resources on GitHub.

This type of authentication gives organization and enterprise owners on GitHub a way to control and secure access to organization resources like repositories, issues, and pull requests. Organization owners can invite GitHub users to join the organization that uses SAML SSO, which allows those users to contribute to the organization and retain their existing identity and contributions on GitHub.

When users access resources within an organization that uses SAML SSO, GitHub will redirect them to the organization's SAML IdP for authentication. After they successfully authenticate with their account on the IdP, the IdP redirects to GitHub to access the organization's resources.

GitHub offers limited support for all identity providers that implement the SAML 2.0 standard with official support for several popular identity providers including:

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

... among others.

### LDAP

Lightweight directory access protocol (LDAP) is a popular application protocol for accessing and maintaining directory information services. LDAP lets you authenticate GitHub Enterprise Server against your existing accounts and centrally manage repository access. It's one of the most common protocols used to integrate third-party software with large company user directories.

GitHub Enterprise Server integrates with popular LDAP services like:

* Active Directory
* Oracle Directory Server Enterprise Edition
* OpenLDAP
* Open Directory

... among others.

# How does GitHub organization and permissions work?

In the previous unit, you explored the different ways that users can authenticate themselves with GitHub. In this unit, you'll learn about permissions for each hierarchical level:

* Repository permissions
* Team permissions
* Organization permissions
* Enterprise permissions

## Repository permission levels

You can customize access to a given repository by assigning permissions. There are five repository-level permissions:

* **Read** - Recommended for non-code contributors who want to view or discuss your project. This level is good for anyone that needs to view the content within the repository but doesn't need to actually make contributions or changes.
* **Triage** - Recommended for contributors who need to proactively manage issues and pull requests without write access. This level could be good for some project managers who manage tracking issues but don't make any changes.
* **Write** - Recommended for contributors who actively push to your project. Write is the standard permission for most developers.
* **Maintain** - Recommended for project managers who need to manage the repository without access to sensitive or destructive actions.
* **Admin** - Recommended for people who need full access to the project, including sensitive and destructive actions like managing security or deleting a repository. These people are repository owners and administrators.

You can give organization members, outside collaborators, and teams different levels of access to repositories owned by an organization. Each permission level progressively increases access to a repository's content and settings. Choose the level that best fits each person or team's role in your project without giving more access to the project than necessary.

## Team permission levels

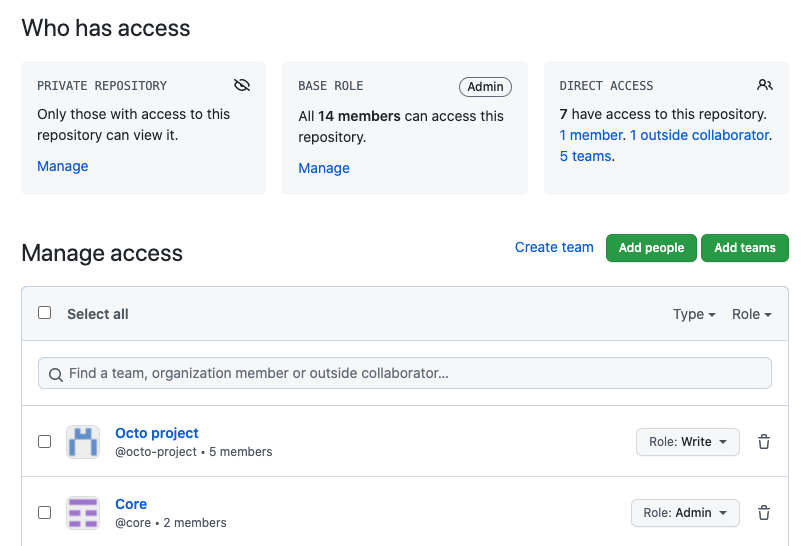
Teams provide an easy way to assign repository permissions to several related users at once. Members of a child team also inherit the permission settings of the parent team, providing an easy way to cascade permissions based on the natural structure of a company.

There are two levels of permissions at the team level:

| **Permission level** | **Description** |
| --- | --- |
| Member | Team members have the same set of abilities as organization members |
| Maintainer | Team maintainers can do everything team members can, plus:  - Change the team's name, description, and visibility  - Request that the team change parent and child teams  - Set the team profile picture  - Edit and delete team discussions  - Add and remove organization members from the team  - Promote team members to also have the team maintainer permission  - Remove the team's access to repositories  - Manage code review assignment for the team  - Manage scheduled reminders for pull requests |

An organization owner can also promote any member of the organization to be a maintainer for a team.

To audit access to a repository that you administer, you can view a combined list of teams and users with access to your repository in your settings:

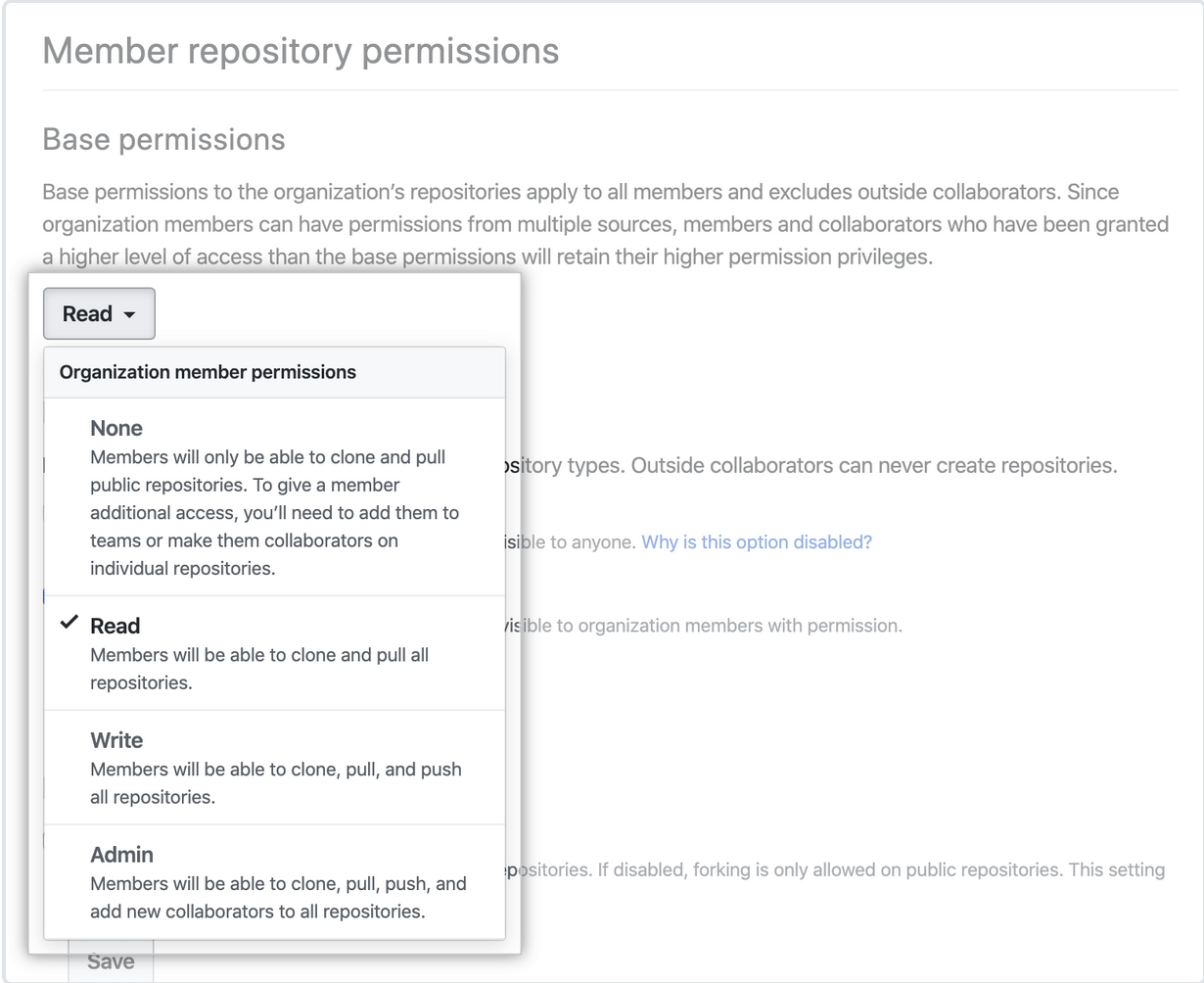


## Organization permission levels

There are three levels of permissions at the organizational level:

| **Permission level** | **Description** |
| --- | --- |
| Owner | Organization owners can do everything organization members can, and can add or remove other users to and from the organization |
| Member | Organization members can create and manage organization repositories and teams |
| Billing manager | Organization billing managers can only view and edit billing information |

In addition to these three levels, you can also set default permissions for all members of your organization:



For improved management and security, you might also consider giving default read permissions to all members of your organization and adjusting their access to repositories on a case-by-case basis. If you have a relatively small organization with a low number of users, a low number of repositories, or a combination of the two, this level of restriction might be unnecessary. If you trust everyone with pushing changes to any repository, you might prefer to give all members write permissions by default.

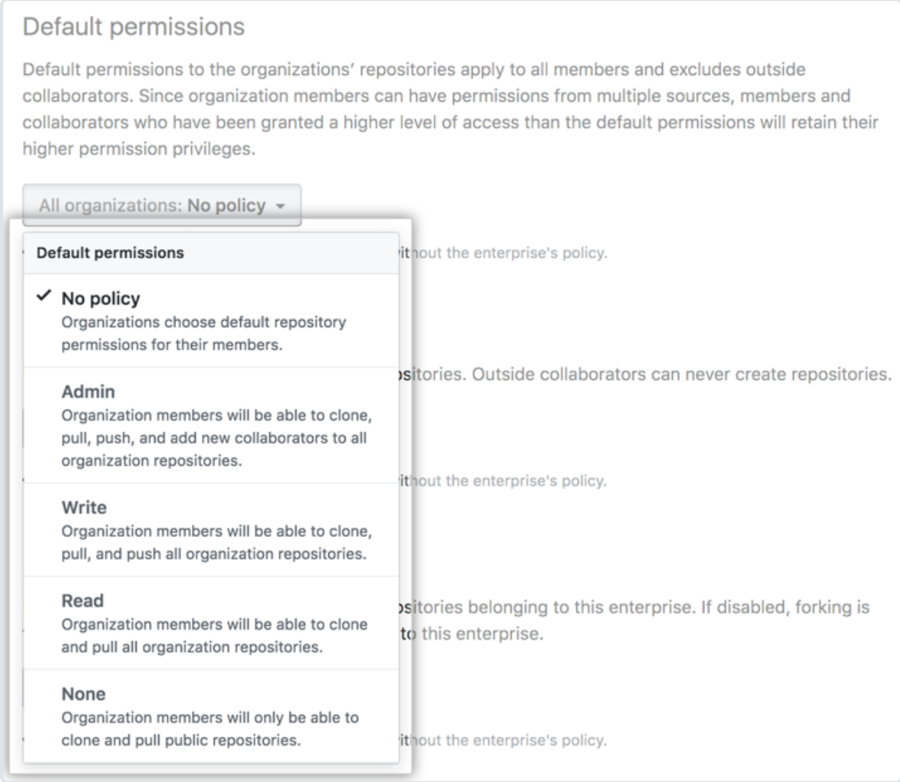
## Enterprise permission levels

Recall from earlier that enterprise accounts are collections of organizations. By extension, each individual user account that is a member of an organization is also a member of the enterprise, and you can control various settings related to authentication from this higher level.

There are three levels of permission at the enterprise level:

| **Permission level** | **Description** |
| --- | --- |
| Owner | Enterprise owners have complete control over the enterprise and can take every action, including:  - Managing administrators  - Adding and removing organizations to and from the enterprise  - Managing enterprise settings  - Enforcing policies across organizations  - Managing billing settings |
| Member | Enterprise members have the same set of abilities as organization members |
| Billing manager | Enterprise billing managers can only view and edit your enterprise's billing information and add or remove other billing managers |

In addition to these three levels, you can also set a policy of default repository permissions across all your organizations:



For improved management and security, you can give default read permissions to all members of your enterprise and adjust their access to repositories on a case-by-case basis. In a smaller enterprise, such as one with a single, relatively small organization, you might prefer to trust all members with write permissions by default.

# Summary

The goal of this module was to help you develop a mental model of the roles and responsibilities of users who perform GitHub administrative tasks for companies.

GitHub supplies administrators with the tools they can use flexibly to control and protect their company's GitHub usage. Administrators can set up authentication schemes and enforce organization-wide or enterprise-wide policies, and design cascading permission structures that represent the natural groupings within the company.

Hierarchical levels like teams, organizations, and enterprises enable ways of setting up and controlling authentication and other security measures. Permission levels allow for fine-grained control of specific tasks. Repository permissions apply to individual users or teams of users, and cascade to child teams.

Without these types of administrative controls, it would be impossible to adequately secure a company's GitHub implementation.

GitHub administrators perform vital tasks that ensure the security and viability of company repositories.

## Learn more

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

* [Organizations](https://docs.github.com/organizations?azure-portal=true)
* [Managing team synchronization for your organization](https://docs.github.com/organizations/managing-saml-single-sign-on-for-your-organization/managing-team-synchronization-for-your-organization?azure-portal=true)
* [Creating a personal access token](https://docs.github.com/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token)
* [Generating a new SSH key and adding it to the ssh-agent](https://docs.github.com/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent?azure-portal=true)
* [Deploy keys](https://docs.github.com/authentication/connecting-to-github-with-ssh/managing-deploy-keys#deploy-keys)
* [Requiring two-factor authentication in your organization](https://docs.github.com/organizations/keeping-your-organization-secure/managing-two-factor-authentication-for-your-organization/requiring-two-factor-authentication-in-your-organization?azure-portal=true)
* [Requiring two-factor authentication for organizations in your enterprise](https://docs.github.com/enterprise-cloud@latest/admin/policies/enforcing-policies-for-your-enterprise/enforcing-policies-for-security-settings-in-your-enterprise#requiring-two-factor-authentication-for-organizations-in-your-enterprise?azure-portal=true)
* [About identity and access management with SAML single sign-on](https://docs.github.com/enterprise-cloud@latest/organizations/managing-saml-single-sign-on-for-your-organization/about-identity-and-access-management-with-saml-single-sign-on?azure-portal=true)
* [Using LDAP](https://docs.github.com/enterprise-server@3.8/admin/identity-and-access-management/using-ldap-for-enterprise-iam/using-ldap?azure-portal=true)
* [Repository roles for an organization](https://docs.github.com/organizations/managing-user-access-to-your-organizations-repositories/repository-roles-for-an-organization#repository-access-for-each-permission-level?azure-portal=true)
* [Managing teams and people with access to your repository](https://docs.github.com/repositories/managing-your-repositorys-settings-and-features/managing-repository-settings/managing-teams-and-people-with-access-to-your-repository?azure-portal=true)
* [Assigning the team maintainer role to a team member](https://docs.github.com/organizations/organizing-members-into-teams/assigning-the-team-maintainer-role-to-a-team-member?azure-portal=true)
* [Roles in an organization](https://docs.github.com/organizations/managing-peoples-access-to-your-organization-with-roles/roles-in-an-organization#permission-levels-for-an-organization?azure-portal=true)
* [Setting base permissions for an organization](https://docs.github.com/organizations/managing-user-access-to-your-organizations-repositories/setting-base-permissions-for-an-organization?azure-portal=true)
* [Roles in an enterprise](https://docs.github.com/enterprise-cloud@latest/admin/user-management/managing-users-in-your-enterprise/roles-in-an-enterprise?azure-portal=true)
* [Enforcing repository management policies in your enterprise](https://docs.github.com/enterprise-cloud@latest/admin/policies/enforcing-policies-for-your-enterprise/enforcing-repository-management-policies-in-your-enterprise#enforcing-a-policy-on-default-repository-permissions?azure-portal=true)