

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 (Chief Information Officer) 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

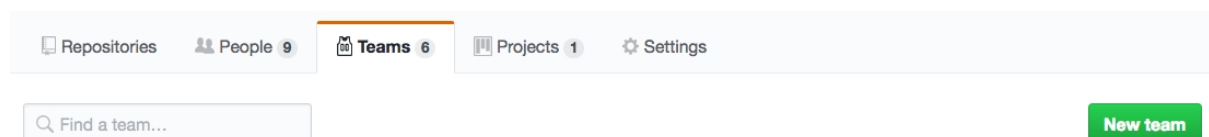
By the end of this module, you'll be able to:

- 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 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 reflecting your company or group's structure. A team is a useful substructure 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 Microsoft Entra ID. When you synchronize a GitHub team with Microsoft Entra ID, you can replicate changes to GitHub automatically. This sync reduces the need for manual updates and custom scripts.

You can use Microsoft Entra ID 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.
- Help streamline PR review processes by creating teams based on interests or specific technology (JavaScript, data science, etc.). Individuals can choose to join these teams according to their interests or skills.
- Enable team synchronization between your 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, you should be aware of the following points:

- Duplicating an organization or sharing configurations between two organizations isn't possible. 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 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 security assertion markup language (SAML) single sign-on for their enterprise account, allowing each enterprise member to link their external identity on your 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.

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, which is 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. We strongly recommend using one (or several) of the other options listed in this unit.

Personal access tokens

Personal access tokens

[Generate new token](#)[Revoke all](#)

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ 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 

[Enable SSO](#)[Delete](#)

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 (SSO). 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

Requiring an additional authentication method adds another level of security for your organization.

☒ **Require two-factor authentication for everyone in the octo-org organization.**

Members and outside collaborators who do not have two-factor authentication enabled for their personal account will be removed from the organization and will receive an email notifying them about the change.

Save

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 2FA for their personal accounts. This action makes 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 IdP, you can configure SAML 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).
- Microsoft Entra ID.
- Okta.
- OneLogin.
- PingOne.

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.

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:

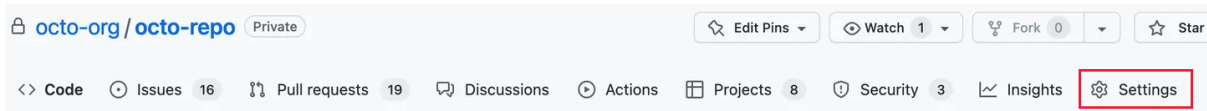
- **Read:** Recommended for non-code contributors who want to view or discuss your project. This level is good for anyone that needs to view the content within the repository but doesn't need to actually make contributions or changes.
- **Triage:** Recommended for contributors who need to proactively 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.

After you create a repository with the correct permissions, you can make it a template so that anyone who has access to the repository can generate a new repository that has the same directory structure and files as your default branch. To make a template:

1. On GitHub.com, go to the main page of the repository.
2. Under the repository name, select **Settings**. If you can't see the **Settings** tab, open the dropdown menu, and then select **Settings**.



3. Select **Template repository**.

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:

Expand table


Permission level	Description
Member	Team members have the same set of abilities as organization members

Maintainer	<p>Team maintainers can do everything team members can, plus:</p> <ul style="list-style-type: none"> - 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:

Who has access

PRIVATE REPOSITORY



Only those with access to this repository can view it.

[Manage](#)

BASE ROLE
Admin

All **14 members** can access this repository.

[Manage](#)


DIRECT ACCESS


7 have access to this repository.
1 member. 1 outside collaborator. 5 teams.


Manage access


[Create team](#)
[Add people](#)
[Add teams](#)

☐ **Select all**
Type ▾ Role ▾


☐


Octo project
 @octo-project • 5 members

Role: Write ▾


☐


Core
 @core • 2 members

Role: Admin ▾


Organization permission levels

There are multiple levels of permissions at the organizational level:

Expand table

Permission level	Description
------------------	-------------

Owner	Organization owners can do everything that organization members can do, and they can add or remove other users to and from the organization. This role should be limited to no less than two people in your organization.
Member	Organization members can create and manage organization repositories and teams.
Moderator	Organization moderators can block and unblock nonmember contributors, set interaction limits, and hide comments in public repositories that the organization owns.
Billing manager	Organization billing managers can view and edit billing information.
Security managers	Organization security managers can manage security alerts and settings across your organization. They can also read permissions for all repositories in the organization.
Outside collaborator	Outside collaborators, such as a consultant or temporary employee, can access one or more organization repositories. They aren't explicit members of the organization.

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

Member repository permissions

Base permissions

Base permissions to the organization's repositories apply to all members and excludes outside collaborators. Since organization members can have permissions from multiple sources, members and collaborators who have been granted a higher level of access than the base permissions will retain their higher permission privileges.

Read

Organization member permissions

None

Members will only be able to clone and pull public repositories. To give a member additional access, you'll need to add them to teams or make them collaborators on individual repositories.

✓ Read

Members will be able to clone and pull all repositories.

Write

Members will be able to clone, pull, and push all repositories.

Admin

Members will be able to clone, pull, push, and add new collaborators to all repositories.

Save

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. You can control various settings related to authentication from this higher level.

There are three levels of permission at the enterprise level:

Expand table

Permission level	Description
Owner	Enterprise owners have complete control over the enterprise and can take every action, including: <ul style="list-style-type: none">- 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:

Default permissions

Default permissions to the organizations' repositories apply to all members and excludes outside collaborators. Since organization members can have permissions from multiple sources, members and collaborators who have been granted a higher level of access than the default permissions will retain their higher permission privileges.

All organizations: No policy ▾

Default permissions

✓ No policy

Organizations choose default repository permissions for their members.

Admin

Organization members will be able to clone, pull, push, and add new collaborators to all organization repositories.

Write

Organization members will be able to clone, pull, and push all organization repositories.

Read

Organization members will be able to clone and pull all organization repositories.

None

Organization members will only be able to clone and pull public repositories.

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.

Repository security and management

You can oversee the security and management of your repositories in several ways.

Create protection rules

To manage changes to content within your repository, you can create [branch protection rules](#) to enforce certain workflows for one or more branches. Protection rules that can be applied to a branch include:

- Require a pull request before merging.
- Require status checks to pass before merging.
- Require conversation resolution before merging.
- Require signed commits.
- Require linear history.

- Require merge queue.
- Require deployments to succeed before merging.
- Lock the branch by making it read-only.
- Restrict who can push to matching branches.

Additionally, you can set branch rules that apply to everyone, including administrators. For example, you can allow force pushes to matching branches and allow deletions from users who have push access.

Add a CODEOWNERS file

By adding a [CODEOWNERS](#) file to your repository, you can assign team members or entire teams as code owners who are responsible for code in the repository. When someone opens a pull request that modifies code that belongs to a code owner, the code owner is automatically requested as a reviewer.

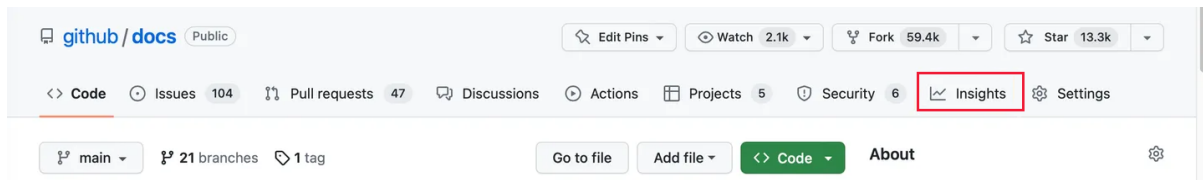
You can create the CODEOWNERS file in either the root of the repository or in the docs or .github folder.

View traffic by using Insights

Anyone who has push access to a repository can [view its traffic](#). In the traffic graph, they can view full clones (not fetches), visitors from the past 14 days, referring sites, and popular content.

To access the traffic graph:

1. On GitHub.com, go to the main page of the repository.
2. Under your repository name, select **Insights**.



3. On the left, select **Traffic**.

Pulse
Contributors
Community
Traffic
Commits
Code frequency
Dependency graph
Network
Forks

4. Optionally, you can select **Clones** or **Views** to see the traffic graph for clones or views

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. They can also 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](#)
- [Managing team synchronization for your organization](#)
- [Managing your personal access tokens](#)
- [Generating a new SSH key and adding it to the ssh-agent](#)
- [Deploy keys](#)

- [Requiring two-factor authentication in your organization](#)
- [Enforcing policies for security settings in your enterprise](#)
- [About identity and access management with SAML single sign-on](#)
- [Using LDAP](#)
- [Repository roles for an organization](#)
- [Managing teams and people with access to your repository](#)
- [Assigning the team maintainer role to a team member](#)
- [Roles in an organization](#)
- [Setting base permissions for an organization](#)
- [Roles in an enterprise](#)
- [Enforcing repository management policies in your enterprise](#)