ConductorOne

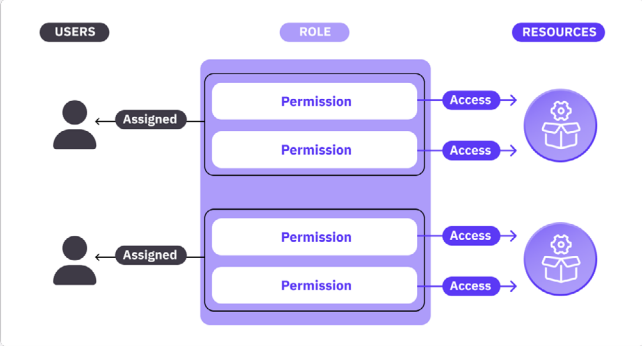
# Everything You Wanted to Know about GitHub Access Control


GitHub is the largest and most popular software development platform, providing services from Git version control to bug tracking, CI/CD, and task management. With over 100 million users developing all kinds of software, proper access control on all levels is crucial to the maintenance, security, and integrity of all the code on the platform. That's why GitHub provides an extensive system to manage access control across repositories, teams, and organizations of all sizes.

This article explores everything you need to know about GitHub access control in order to properly manage your GitHub accounts and repositories on all levels.

## Understanding GitHub's Authorization/Permission Model

GitHub uses a role-based access control (RBAC) model. With this model, to perform any action on GitHub, a user must have appropriate permissions that grant them the necessary access to a particular resource at the given level. Permissions are collectively assigned to users as roles:




There are different roles for different types of GitHub accounts. For a repository owned by a personal account, the only two available roles are owner and collaborator:

- An owner has complete control over the repository, with the ability to perform dangerous, destructive actions (like archiving or deleting the repository) and add collaborators.
- A collaborator has read/write access to the repository's content, with the ability to manage issues, merge pull requests, and create releases, for example.

If you need more granular control, you have to create a GitHub organization, where you'll have access to more organization- and repository-level roles.

At the organizational level, the following roles are available:

- An owner has complete control over the organization, including all of its repositories and assigned users. To ensure the organization's continuity, it's recommended that it have at least two owners.
- A member is the base, non-administrative role with default access to repos and projects; its permissions are configurable.
- A moderator has extended moderation-related permissions in addition to base member role permissions, like blocking non-member users, managing interaction limits, and hiding disruptive comments in public repos.
- The billing manager role can manage the organization's billing settings, such as the current plan, sponsorships, payment details, or history.

- The GitHub App manager role can manage some or all GitHub Apps registered by the organization on top of the basic, required member role's permissions.
- The outside collaborator role allows limited access to selected organization repositories. It has to be further controlled with repository-level permissions.

You can also configure the permissions of individual members, outside collaborators, and entire teams in relation to selected repositories by assigning them one of the repository-level roles:

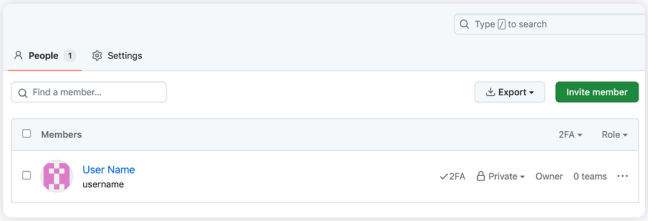
- **Read** provides read-only access to the repo's content and related resources (issues, pull requests, and so on).
- **Triage** has additional permissions to better proactively manage issues, pull requests, and more.
- **Write** has further write permissions to the repo's content to perform actions such as merging pull requests or creating releases.
- **Maintain** has broad access to the repository, with the exception of dangerous or destructive actions.
- **Admin** has full access to the repository, including permissions to change security settings or delete the repository.

## Managing Authorization and Access Control in GitHub Organizations

If you have the necessary permissions, you can manage the roles of other users in the dedicated settings panels of the entire organization or individual repo. If you want to do this effectively, you'll need a strong understanding of GitHub's access control system and its UI dashboard.

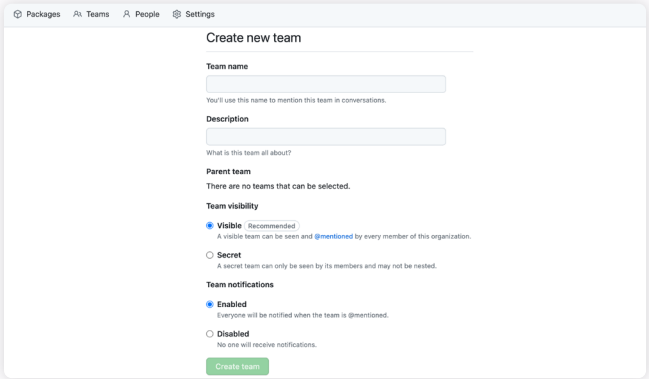
### General User and Team Management

Managing your GitHub organization begins with inviting members. From the main organization dashboard, go to the **People** tab and click **Invite member**. Then, from the respective dropdown menu, you can convert any member to an outside collaborator, further manage their access to select repositories, or completely remove them from the organization:

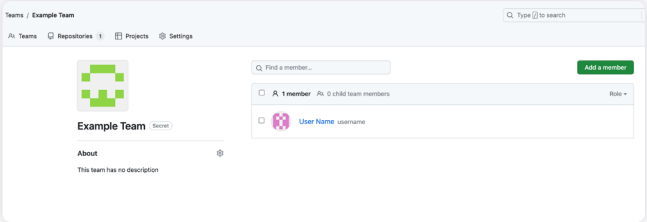



In addition to managing members individually, you can also assign them to teams and manage their repository access and permissions collectively.

To create a new team, head to the **Teams** tab and click **New team**, then fill in the relevant details:

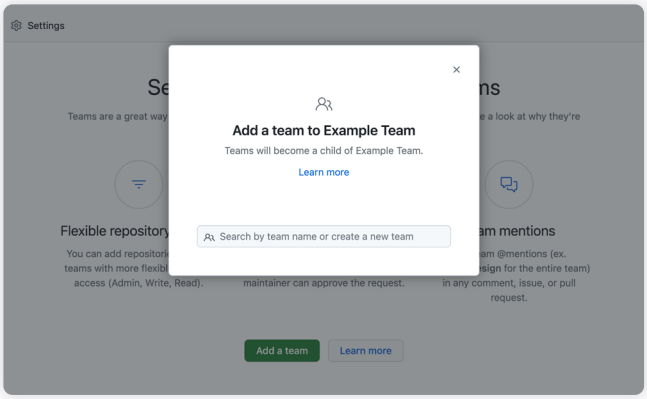


Once you create a team, you'll be redirected to its dedicated dashboard:

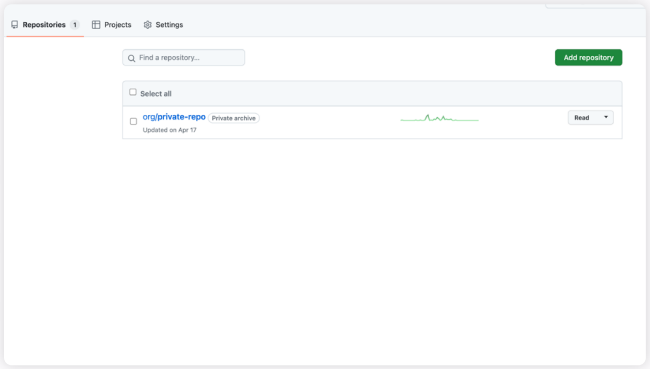



From here, you can manage the team, including its details, members, and repository-level roles. You can add a member to the team by clicking **Add a member**. This can be someone who is already an organization member or someone you are inviting to become a member of both the team and the entire organization.

In the **Teams** tab, you can also add an entire child team by clicking **Add a team**. This can be used to better organize and manage access control:




You can assign a repository-level role to a team to provide repository access. To do this, go to the **Repositories** panel and click **Add repository**:

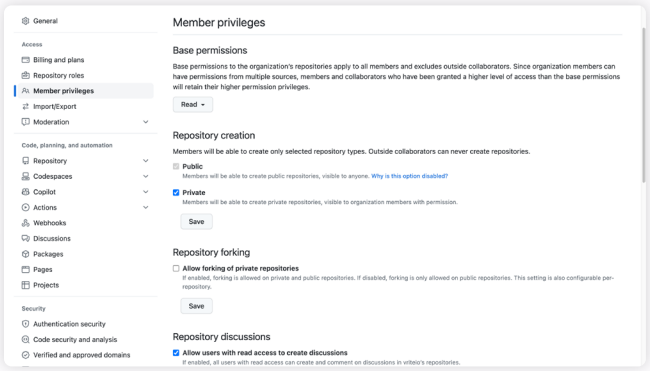



For each repository you add, you'll have to choose a role to assign to the team.

## Organization-Level Access Control

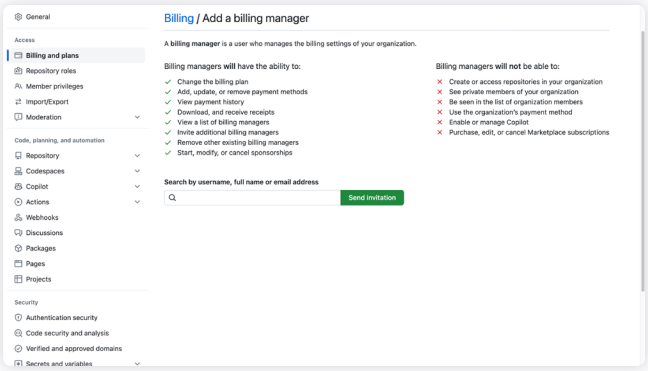
The member role serves as the base role for the organization's users. GitHub provides extra organization-level tools to enhance control over the privileges associated with this role.

You can configure various options for the member role in the organization's **Settings** tab under the **Member privileges** section. Most importantly, this includes base permissions, meaning the level of access each member will possess across all repositories. Additionally, you can customize access to various actions within repositories, projects, issues, and more:

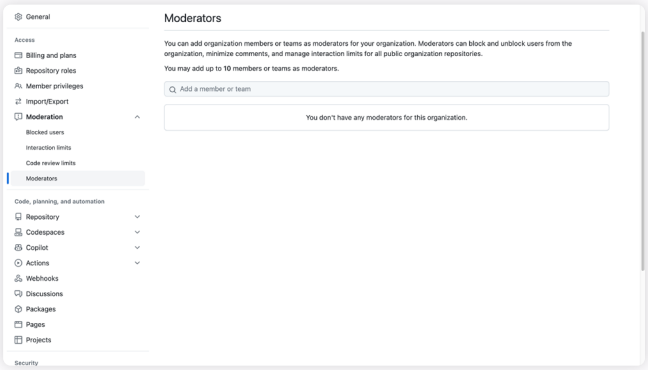


You can also assign other organization-level roles from the **Settings** tab.

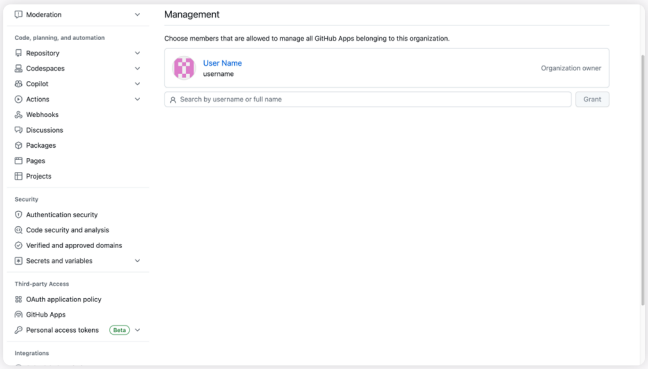
To assign billing managers, go to the **Billing and plans > Billing managers** section and click **Invite**:



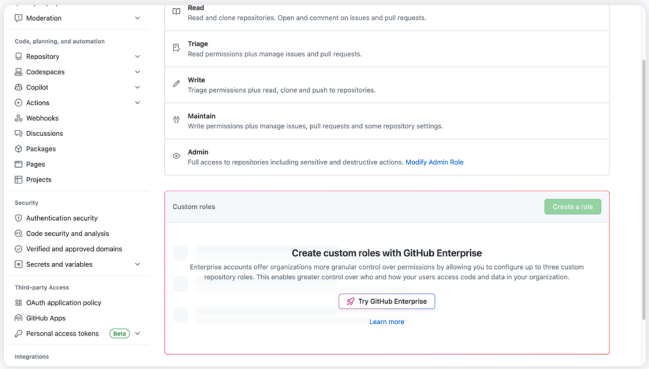

You can assign existing users or entire teams as moderators in the **Moderation > Moderators section**:




Lastly, you can assign the GitHub App manager role to existing organization members from the **Developer Settings > GitHub Apps > Management** section:



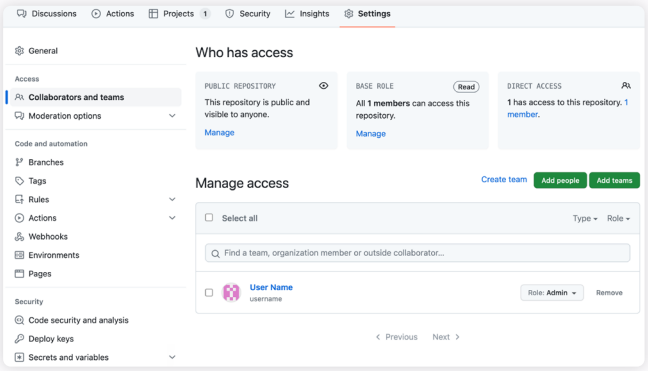

Among the different settings for managing your organization, the **Repository roles** section is particularly useful. It lets you see all the predefined repository-level roles. If you have a GitHub Enterprise account, it also lets you create custom repository roles for even more fine-grained control:



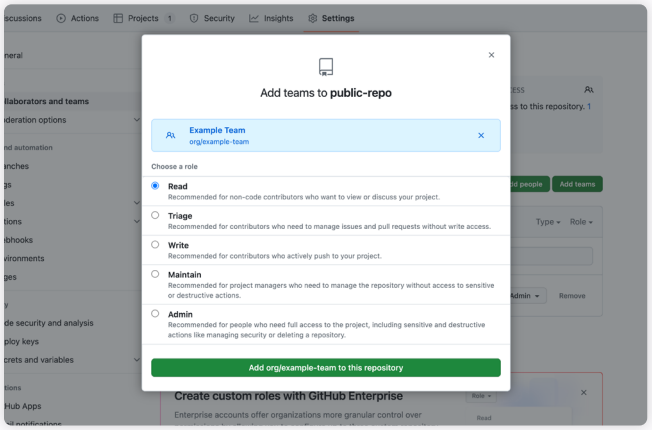

Custom repository roles aren't the only advantage of the GitHub Enterprise plan when it comes to access control. The plan also gives you access to more authentication security settings, including SAML Single Sign-On (SSO), custom enterprise policies, and an IP allow list—all certified for compliance.

## Repository-Level Access Control

In addition to configuring repository-level roles through settings for members or teams, you can also do this at the level of a given repository:




In the repository dashboard, under **Settings > Collaborators and teams**, you can use the **Add people** and **Add teams** buttons and manage repository-level roles accordingly for members and outside collaborators or teams:




This provides you with an alternative way to manage access control, reassuring you that the changes you make affect only the selected repository.

## Strengths of GitHub's Authorization Model

Having a good understanding of GitHub's access control model, its implementation, and its usage allows you to better understand its strengths and limitations.

Having granular controls and the ability to configure individual user access to selected repositories allows you to manage an organization at any scale. GitHub's RBAC model, its built-in roles, and its implementation of teams and sub-teams provide enough control and flexibility so that you can model the access control settings after the way your organization is structured.

Large organizations with hundreds of members can use teams and sub-teams to manage general access at the team level. Individual teams can then manage their own members, allowing for easier control within a focused scope. On the other hand, smaller organizations can manage individual members' roles directly and take advantage of the "outside collaborator" role for onboarding new or temporary members.

Further access control configuration at the repository level facilitates collaboration and various repository-bound workflows that are available on GitHub. From managing pull requests and issues to using GitHub projects or CI/CD, proper access control makes sure only trusted members have access to dangerous or destructive actions.

## Limitations and Challenges of GitHub's Authorization/Permission Model

That said, GitHub's model does have its limitations.

GitHub's RBAC implementation provides many tools that allow you to have granular control over the roles of every member of the organization. However, with such flexibility comes increased complexity, which can quickly become a challenge, especially when managing a large organization. From the organization-level roles, through teams and sub-teams, down to individual members' permissions to select repositories, having to manage and delegate tasks through all the levels can quickly get out of hand.

While complexity may only affect large organizations, the limitations of GitHub's RBAC implementation can be experienced at any scale. Even though GitHub has a concept of permissions, you're limited to assigning predefined roles for the most part. The few configuration options GitHub provides may not be enough in case you need to create custom roles (for example, for contractors or community managers) to accurately model access control in your organization. Such a level of control is reserved only for GitHub Enterprise accounts.

However, custom roles aren't the only advantage of GitHub Enterprise Cloud. If your organization has to adhere to special compliance and security standards, no account type other than Enterprise can fit your needs. This vastly limits your options and can be especially problematic for smaller, specialized organizations, which must adhere to various standards but have little need for other features of GitHub Enterprise. Still, in such cases, Enterprise Cloud, being certified for compliance (with annual SOC 1 Type 2 and SOC 2 Type 2 reports) and providing extended authentication security features (like SAML SSO), is your only option.

## Conclusion

You should now know how GitHub's RBAC model works and how you can make the best use of it in your GitHub organization.

Implementing role-based access control as advanced as GitHub's might seem challenging, but it doesn't have to be.

With ConductorOne, an identity security automation platform, you can easily add modern, user-focused access controls with streamlined self-service requests and just-in-time provisioning to any app. Check out the official documentation to learn more.