ConductorOne

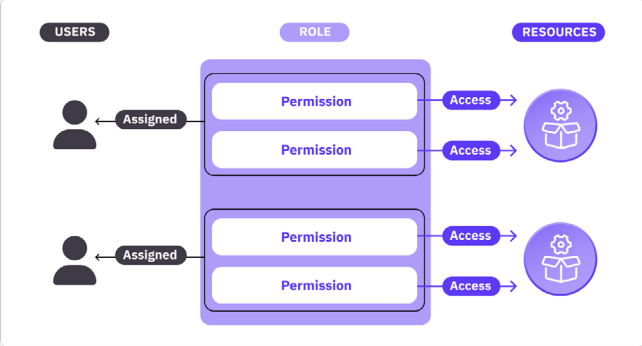
# Everything You Wanted to Know about GitHub Access Control


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:

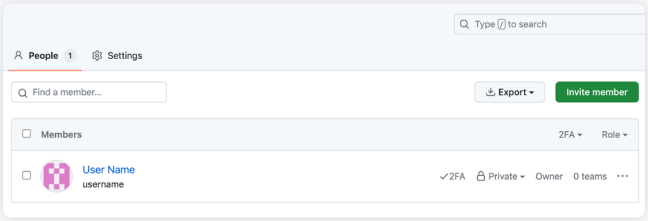
- **Read** provides read-only access to the repo's content and related resources (issues, pull requests, and so on).
- **Triage** has additional permissions to better proactively manage issues, pull requests, and more.
- **Write** has further write permissions to the repo's content to perform actions 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.

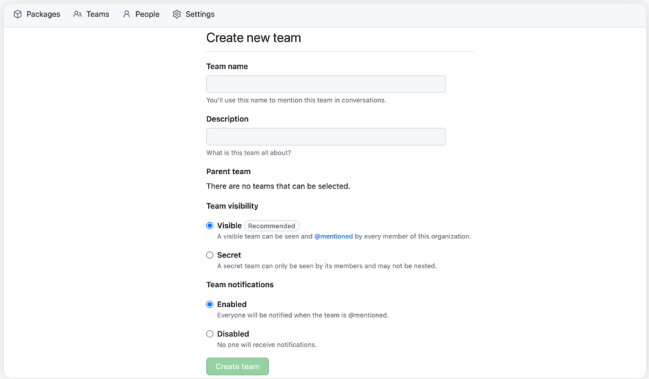
### General User and Team Management

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:

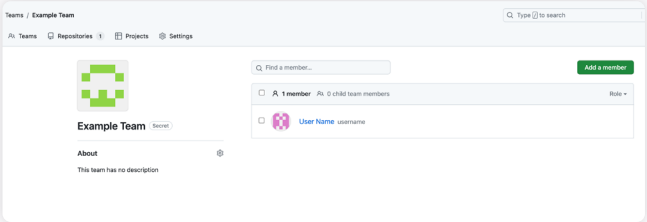



In addition to managing members individually, you can also assign them to teams and manage their repository access and permissions collectively.

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

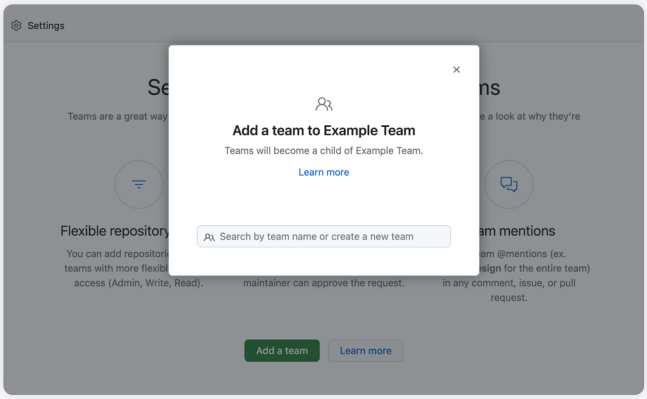


Once you create a team, you'll be redirected to its dedicated dashboard:

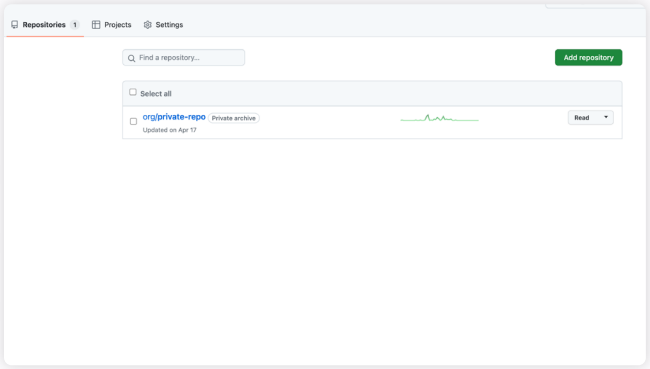



From here, you can manage the team, including its details, members, and repository-level roles. You can add a member to the team by clicking **Add a member**. This can be someone who is already an organization member or someone you are inviting to become a member of both the team and the entire organization.

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: