# About custom repository roles

**In this article**

You can more granularly control access to your organization's repositories with custom repository roles.

## About custom repository roles

To perform any actions on GitHub Enterprise Server, such as creating a pull request in a repository or changing an organization's billing settings, a person must have sufficient access to the relevant account or resource. This access is controlled by permissions. A permission is the ability to perform a specific action. For example, the ability to delete an issue is a permission. A role is a set of permissions you can assign to individuals or teams.

Within an organization, you can assign roles at the organization, team, and repository level. For more information about the different levels of roles, see "Roles in an organization."

You can have more granular control over the permissions you grant at the repository level by creating up to five custom repository roles. A custom repository role is a configurable set of permissions with a custom name you choose. For more information, see "Managing custom repository roles for an organization."

After you create a custom role, anyone with admin access to a repository can assign the role to an individual or team. For more information, see "Managing an individual's access to an organization repository" and "Managing team access to an organization repository."

You can also use the REST API to create and manage custom repository roles. For more information, see "Custom Repository Roles."

## About the inherited role

When you create a custom repository role, you start by choosing an inherited role from a set of pre-defined options. The inherited role determines the initial set of permissions included in the custom role. Then, you can further customize the role by choosing additional permissions to give the role. For the full list of available permissions, see "Additional permissions for custom roles."

Your options for the inherited role are standardized for different types of contributors in your repository.

| Inherited role | Designed for |
|---|---|
| **Read** | Non-code contributors who want to view or discuss your project |
| **Triage** | Contributors who need to proactively manage issues and pull requests without write access |
| **Write** | Organization members and collaborators who actively push to your project |
| **Maintain** | Project managers who need to manage the repository without access to sensitive or destructive actions |

## Custom role examples

Here are some examples of custom repository roles you can configure.

| Custom repository role | Summary | Inherited role | Additional permissions |
|---|---|---|---|
| Security engineer | Able to contribute code and maintain the security pipeline | **Maintain** | Delete code scanning results |
| Contractor | Able to develop webhooks integrations | **Write** | Manage webhooks |
| Community manager | Able to handle all the community interactions without being able to contribute code | **Read** | - Mark an issue as duplicate<br>- Manage GitHub Page settings<br>- Manage wiki settings<br>- Set the social preview<br>- Edit repository metadata<br>- Triage discussions |

## Additional permissions for custom roles

After choosing an inherited role, you can select additional permissions for your custom role.

You can only choose an additional permission if it's not already included in the inherited role. For example, if the inherited role offers **Write** access to a repository, then the "Close a pull request" permission will already be included in the inherited role.

### Discussions

- Create a discussion category
- Edit a discussion category
- Delete a discussion category
- Mark or unmark discussion answers
- Hide or unhide discussion comments
- Convert issues to discussions

For more information, see "GitHub Discussions documentation."

### Issue and Pull Requests

- Assign or remove a user
- Add or remove a label

### Issue

- Close an issue
- Reopen a closed issue
- Delete an issue
- Mark an issue as a duplicate

### Pull Request

- Close a pull request
- Reopen a closed pull request
- Request a pull request review

### Repository

- Set milestones
- Manage wiki settings
- Manage project settings
- Manage pull request merging settings
- Manage GitHub Pages settings (see "Configuring a publishing source for your GitHub Pages site")
- Manage webhooks
- Manage deploy keys
- Edit repository metadata
- Set the social preview
- Push commits to protected branches
  - Base role must be `write`
  - Branch protection rules will still apply
- Create protected tags
- Delete protected tags
- Bypass branch protections
- Edit repository rules

### Security

- View code scanning results
- Dismiss or reopen code scanning results
- Delete code scanning results
- View Dependabot alerts
- Dismiss or reopen Dependabot alerts
- View secret scanning results
- Dismiss or reopen secret scanning results

## Precedence for different levels of access

If a person is given different levels of access through different avenues, such as team membership and the base permissions for an organization, the highest access overrides the others. For example, if an organization owner gives an organization member a custom role that uses the "Read" inherited role, and then an organization owner sets the organization's base permission to "Write", then this custom role will have write access, along with any additional permissions included in the custom role.

If a person has been given conflicting access, you'll see a warning on the repository access page. The warning appears with "⚠ Mixed roles" next to the person with the conflicting access. To see the source of the conflicting access, hover over the warning icon or click **Mixed roles**.

To resolve conflicting access, you can adjust your organization's base permissions or the team's access, or edit the custom role. For more information, see:

- "Setting base permissions for an organization"
- "Managing team access to an organization repository"
- "Editing a repository role"