|  |
| --- |
| The Auto Club Group |
| GitHub Enterprise Process / Policy |
|  |
|  | |



Contents

[Enterprise GitHub 3](#_Toc10620912)

[Ownership of the Application / Infrastructure 3](#_Toc10620913)

[How it works – In a Nutshell 3](#_Toc10620914)

[Creation of a Project Repository 4](#_Toc10620915)

[User Roles 4](#_Toc10620916)

[Owners 4](#_Toc10620917)

[Members 4](#_Toc10620918)

[Repository permission levels for an organization 5](#_Toc10620919)

[Requesting Access to the Application / Project 6](#_Toc10620920)

[Request to be added to AD Group 6](#_Toc10620921)

[Assignment to Repository 7](#_Toc10620922)

[Enterprise Standards 7](#_Toc10620923)

[Branching Model 8](#_Toc10620924)

# Enterprise GitHub

Enterprise GitHub hosts Git repositories located on premise that has many features. While Git is a command line tool, GitHub provides a Web-based and a Desktop Client graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

ACG is moving away from Subversion (SVN) for better control and code sharing among the entire enterprise. GitHub also has better interactions with LDAP and Active Directory for better user administration. This section of the document will discuss the following:

1. Ownership of the Application / Infrastructure
2. How it works – In a Nutshell
3. Creation of a Project
4. User Roles:
   1. Owner
   2. Member
5. Repository permission levels for an organization
6. Requesting Access to the Application / Project
   1. Request to be added to AD Group
   2. Assignment to Repository

## Ownership of the Application / Infrastructure

GitHub is owned and administered by the Configuration Management department with the responsibility to onboard users, create teams, assign users to teams, and assign teams to repositories.

## How it works – In a Nutshell

GitHub server has been installed and joined to the Network. Two Active Directory groups have been created in which users have been assigned; the GitHub-Admin Group and the GitHub-User Group.

Having the users added to the two LDAP security group only allows them certain access to the GitHub server. The Admin group gives administrators full rights to the application on the software level and the ability to create and remove repositories, add and remove users, and administer the backend of the application and configuration; whereas the user group only allows access to the application itself.

All users who are assigned to the GitHub server will be required to login at least once either by using the desktop client or the web interface. By logging into the server, the user account is then registered to the GitHub server.

## Creation of a Project Repository

Project repository creation can only be made by the users which are assigned to the GitHub-Admin Group. As an admin, the user will have the capability to create the repository and assign users. Moreover, the admin also has the capability to make the repository private ( default ) or public to the entire enterprise. Once a repository is created, leads can be assigned as Maintainers of the repository.

## User Roles

After the organization and project are create, the administrator / creator of the project and repository should then give Owner permissions to a small group of people who will manage the organization account.

NOTE: Organization members can have owner or member roles

### Owners

Owners have complete administrative access to your organization. This role should be limited to a few people in your organization. For more information, see "Changing a person's role to owner."

### Members

Members are the default role for everyone else.

See GitHub Organization Action Chart below:

|  |  |  |
| --- | --- | --- |
| **Organization action** | **Owners** | **Members** |
| Invite people to join the organization | X |  |
| Edit and cancel invitations to join the organization | X |  |
| Remove members from the organization | X |  |
| Reinstate former members to the organization | X |  |
| Add and remove people from all teams | X |  |
| Promote organization members to team maintainer | X |  |
| Add collaborators to all repositories | X |  |
| Access the organization audit log | X |  |
| Delete all teams | X |  |
| Delete the organization account, including all repositories | X |  |
| Create teams | X | X |
| See all organization members and teams | X | X |
| @mention any visible team | X | X |
| Can be made a team maintainer | X | X |
| Transfer repositories | X |  |

## 

## Repository permission levels for an organization

Users with owner or team maintainer roles can manage repository access with teams. Each team can have different repository access permissions.

There are three types of repository permissions available for people or teams collaborating on repositories that belong to an organization:

* Read
* Write
* Admin

In addition, organization members with owner permissions have extensive permissions across all repositories in an organization. See below chart for Repository Actions and Permissions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Repository action** | **Permissions** | | | |
| **Read** | **Write** | **Admin** | **Owner** |
| Pull (read), push (write), and clone (copy) all repositories in the organization |  |  |  | X |
| Promote organization members to team maintainer |  |  |  | X |
| Convert organization members to outside collaborators |  |  |  | X |
| Create repositories (see "Creating repositories" for details) | X | X | X | X |
| Delete repositories (see "Deleting repositories" for details) |  |  | X | X |
| Transfer repositories into the organization account (see "Creating repositories" for details) |  |  | X | X |
| Transfer repositories out of the organization account (see "Deleting repositories" for details) |  |  | X | X |
| Change a repository's settings (see "Changing repository settings" for details) |  |  | X | X |
| Change a repository's visibility |  |  | X | X |
| Add a repository to a team (see "Adding a repository to a team" for details) |  |  | X | X |
| Add outside collaborators to a repository |  |  | X | X |
| Remove outside collaborators from a repository |  |  | X | X |
| Pull from (read) the team's assigned repositories | X | X | X | X |
| Push to (write) the team's assigned repositories |  | X | X | X |
| Fork (copy) the team's assigned repositories | X | X | X | X |
| Send pull requests from forks of the team's assigned repositories | X | X | X | X |
| Merge and close pull requests |  | X | X | X |
| Merge pull requests on protected branches, even if there are no approved reviews |  |  | X | X |
| Submit reviews on pull requests | X | X | X | X |
| Submit reviews that affect a pull request's mergeability |  | X | X | X |
| Open issues | X | X | X | X |
| Close, reopen, and assign issues |  | X | X | X |
| Close issues they opened themselves | X | X | X | X |
| Apply labels and milestones |  | X | X | X |
| Have an issue assigned to them | X | X | X | X |
| Create and edit releases |  | X | X | X |
| View draft releases |  | X | X | X |
| View published releases | X | X | X | X |
| Edit and delete their own comments on commits, pull requests, and issues | X | X | X | X |
| Edit and delete anyone's comments on commits, pull requests, and issues |  | X | X | X |
| Edit wikis | X | X | X | X |
| Create statuses |  | X | X | X |
| Create project boards |  | X | X | X |

## Requesting Access to the Application / Project

Since all current SVN users have been migrated to GitHub and ACG Security Admin had already transferred the user ID’s to the Active Directory Group, any future requests to the GitHub Server and groups will need to be requested and added through the [Identity Self Service](https://caas.acg.aaa.com/identity). As stated above in the [User Role section](#_User_Roles), once the users are assigned to the LDAP Group, the assigned manager to the repository can then assign the users as a member to the repository. To request access to either the GitHub-Admin or the GitHub-User group, the user or the manager must follow the steps outlined below:

### Request to be added to AD Group

1. Logon to the [ACG Hub](http://hub.autoclubgroup.com/pages/home.aspx) – If you do not have access to the Hub, please see step two which will take you directly to the Identity Self Service site.
2. From the Application Launchpad, Click on [Identity Self Service](https://caas.acg.aaa.com/identity) (ISS)
3. Once at the Page, Enter P-ID and Password and Click Login
4. Once in the ISS, click on the box labeled Request Access:
   1. Select Request for Self if you are requesting access for yourself
   2. Select Request for Others if you are requesting access for someone on your team
5. Once the page opens, run a search for Git
6. You will be presented with choices:
   1. GitHub-Admin – If requesting to be in the admin group
   2. GitHub-User – If you are requesting to be added as a member to the GitHub Server
7. Once you select the group, Click on Add to Cart
8. Click the Next Button on top of the Page
9. Click on Submit to send in the request

### Assignment to Repository

Once the [Request to be added to AD Group](#_Request_to_be) has been submitted, approved, and implemented, the user will be required to login to the GitHub Server in order to register the account. Upon successful login, the user should inform the assigned repository manager so he or she may assign the user to a repository.

## Enterprise Standards

* Repositories will be created under ACG organization
* Each application / service will have its own repository
* LDAP will be used for authentication, however teams will be maintained in GitHub
* Repository naming convention
  + Snake case will be used
  + Descriptive names will be used to identify the application and functional area
* Follow Branch Modeling
  + Master – main branch reflecting the latest development changes. Release from Master Branch only!
  + Developers should branch and merge from Master
  + Supporting Branches
    - Feature – for new features / enhancements, must branch and merge with master
    - Bug – for bug fixes, similar to Feature but short-lived
    - Hotfix – fix undesirable Production state

## Branching Model

























