

Git and GitHub

❖ “*GitHub Organization – A unified hub for teams to securely collaborate, manage, and ship code at scale.*”

Description:

GitHub Organization is a collaborative workspace that brings together people, teams, and repositories under one umbrella. It enables centralized management of projects, user permissions, security policies, and workflows, making it ideal for teams and enterprises to build, review, and ship code efficiently and securely.

- **GitHub Organizations**
- **GitHub Repository's permissions**
- **GitHub Branches protection**
- **Collaborators (Give the Specific user permissions)**
- **Pull Requests (Branching & Merging)**

These are Some Specific Features/Toggles in the GitHub Organization:

- ❖ **Organization roles**
 1. Role management
 2. Role Assignment
- ❖ **Repository Roles**
- ❖ **Member Privileges**
- ❖ **Moderation**
 1. Block Users
 2. Interaction Limits
 3. Code Review Limits
 4. Moderators
- ❖ **Two Factor Authentication**

GitHub Organization:

GitHub and manage its privacy:

➔ Create a New Organization:

- Log in to your GitHub account.
- Click your profile photo in the upper-right corner of any page, then click Settings.

- In the "Access" section of the sidebar, click Your Organizations.
- Click New organization next to the "Organizations" header.
- Follow the prompts to create your organization, including choosing a plan (GitHub Free, Team, or Enterprise).

→ Add users to the org with **role-based permissions**:

- **Read** → can only view code (still allows download unless combined with other controls).
- **Write** → can push commits.
- **Triage** → Manage issues/PRs without write.
- **Maintain** → repo settings (without admin).
- **Admin** → full control.
- Use **Teams** (e.g., *Dev, QA, Ops*) with repo-level permissions.
- Restrict **production repo** to *Admin + Ops team only*. Developers can be given access only to dev/staging repos.

→ Prevent Direct Zip/Clone Access

- By default, if a user has **read** access, they can download the repo as .zip or clone it.
- To block this, you must **avoid giving read access at all**.

Alternatives:

- **Private repositories**: Only explicitly added members can access.
- **Deploy Keys / GitHub Apps**: Give your CI/CD pipeline access without letting developers download the repo directly.
- **Code Owners**: Enforce code reviews before merges.
- **Branch Protection Rules**: Prevent force-push, require PR reviews, and restrict who can merge into production branches.

Note: GitHub does not provide a direct "disable zip download" toggle. If someone has read access, they can always get the code.

Workaround:

- Keep the **production repo private**.
- Developers work only on **mirror repos** or **fork of staging**.
- Only CI/CD pipelines (via tokens/keys) pull from the **production repo** to deploy.

→ Manage Repository Visibility:

- Within your organization, you can create repositories and set their visibility to Private. This ensures that only members of your organization, or specific teams within it, can access the repository's code and content.

→ Control Member and Team Visibility:

- You can choose to hide your individual membership within an organization by setting your visibility to Private in your

organization's "People" settings.

- You can also manage the visibility of teams within your organization, controlling who can see and interact with them.

→Fine-grained Access Control with Teams:

- For private repositories within your organization, you can create teams and grant specific access levels (e.g., read, write, admin, maintain, triage) to those teams on a per-repository basis. This allows for precise control over who can access and contribute to your private projects.

By combining these features, you can create a highly secure and private environment for your projects and collaborations within a GitHub Organization

Step-By-Step Process:

The screenshot shows the GitHub 'Choose a plan' page at <https://github.com/organizations/plan>. The page title is 'Choose a plan for your organization'. It displays three plan options: 'Free' (\$0 USD per user/month), 'Team' (\$4 USD per user/month), and 'Enterprise' (\$21 USD per user/month). Each plan box lists its features. A 'Compare features' button is located below the plan boxes. At the bottom, there are 'Join for free', 'Continue with Team', and 'Start Enterprise Trial' buttons.

Plan	Cost	Features
Free	\$0 USD per user/month	Unlimited public/private repositories Dependable security and version updates 2,000 CI/CD runs/month (not included in public repository limit) Issues & Projects Community support GitHub API v3.5 GitHub Copilot Access GitHub CodeReview Access
Team	\$4 USD per user/month	Advanced collaboration for individuals and organizations Everything included in Free plus... Access to GitHub Packages Repository rules Multiple reviewers in pull requests Dish poll requests Code owners Required reviewers Pages and Wikis Environment deployment branches and assets 200 Packages storage (not included in public repository limit) Web-based support GitHub API v3.5 GitHub Secret Protection GitHub Code Security
Enterprise	\$21 USD per user/month	Everything included in Team plus... Basic monitoring Enterprise Managed Users User provisioning through SCIM Enterprise Account to centrally manage multiple organizations Enhanced protection rules Repository rules Audit Log API SOC2, ISO27001, Type 2 audits annually FedRAMP General Authority to Operate (GAO) SSAK single sign-on Additional auditing GitHub Connect 2,000 CI/CD runs/month (not included in public repository limit) 50GB of Package storage (not included in public repository limit) Premium support

→Here Select your organization it may be free, Team, Enterprise level

→In the Enterprise level we have two types on-premises & off-premises Deployment

On-premises: In these we have GitHub website is going to use or GitHub Cloud.

Off-premises: In these we have to use Physical Resources it may be physical servers or any other else.

→Go to Organizations create new organization

The image displays three screenshots of a web browser showing the process of creating a new organization on GitHub.

Screenshot 1: GitHub Settings - Organizations

This screenshot shows the GitHub Settings page for the user "Venkat-Singareddy". The left sidebar has "Organizations" selected. The main area displays the message "You are not a member of any organizations." Below this, there's a section titled "Transform account" with a button "Turn Venkat-Singareddy into an organization".

Screenshot 2: GitHub Account - Set up your organization

This screenshot shows the "Set up your organization" step. It asks for the organization name ("GenAllLakes099") and contact email ("singareddy099@gmail.com"). It also asks if the organization belongs to "My personal account" or "A business or institution". A CAPTCHA challenge is shown below. The URL in the address bar is [https://github.com/account/organizations/new?plan=free&ref_cta/Create%2520a%2520free%2520organization&ref_loc=cards&ref_p...](https://github.com/account/organizations/new?plan=free&ref_cta>Create%2520a%2520free%2520organization&ref_loc=cards&ref_p...).

Screenshot 3: GitHub Organization - Welcome to GenAllLakes099

This screenshot shows the newly created organization "GenAllLakes099". It prompts the user to "Start collaborating" and "Welcome to GenAllLakes099". It includes a section for "Add organization members" with a note about permissions and a search bar for users. A green "Complete setup" button is at the bottom. The URL in the address bar is <https://github.com/organizations/GenAllLakes099/invite>.

The screenshot shows the GitHub organization 'GenAllLakes099' overview page. At the top, there's a 'Confirm access' dialog box. Below it, the organization's profile card displays the name 'GenAllLakes099' and a purple logo. The main content area features a heading 'We think you're gonna like it here.' followed by several call-to-action cards:

- Invite your people**: 'Invite your first member' and 'Customize members' permissions'.
- Collaborative coding**: 'Create a pull request' and 'Create a branch protection rule'.
- Automation and CI/CD**: 'Auto-assign new issues' and 'Run a continuous integration test'.

On the right side of the page, there are sections for 'Discussions', 'Repositories', and 'People'. A sidebar on the left lists organization settings like Overview, Repositories, Projects, Packages, AI Teams, People, Insights, and Settings.

❖ Organization roles

1. Role management

→ In Your Organization you can assign Role Management

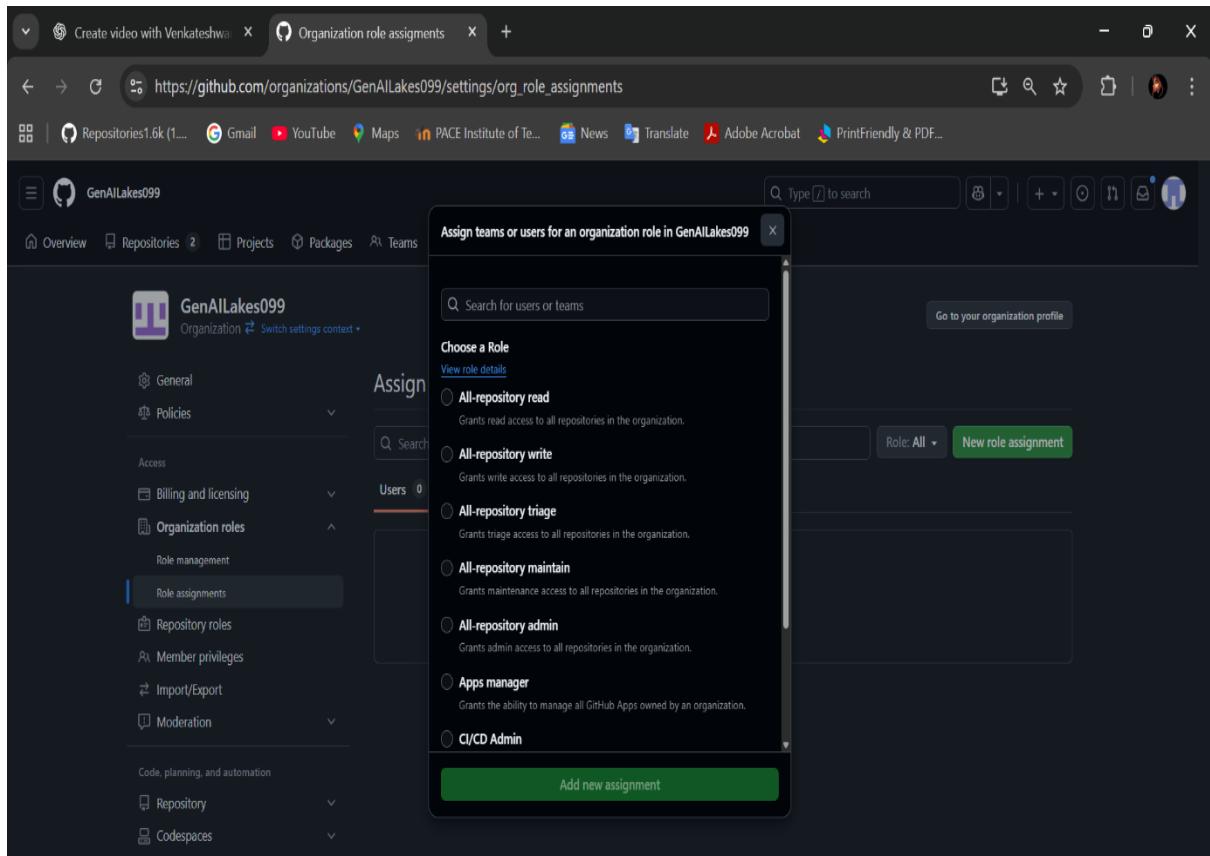
The screenshot shows the 'Role management' section of the GitHub organization settings. On the left, there's a sidebar with navigation links like General, Policies, Access, Billing and licensing, Organization roles, Role assignments, Repository roles, Member privileges, Import/Export, and Moderation. The 'Organization roles' link is currently selected. The main content area is titled 'Role management' and contains a list of repository roles:

- All-repository read
- All-repository write
- All-repository triage
- All-repository maintain
- All-repository admin
- Apps manager
- CI/CD Admin
- Security manager

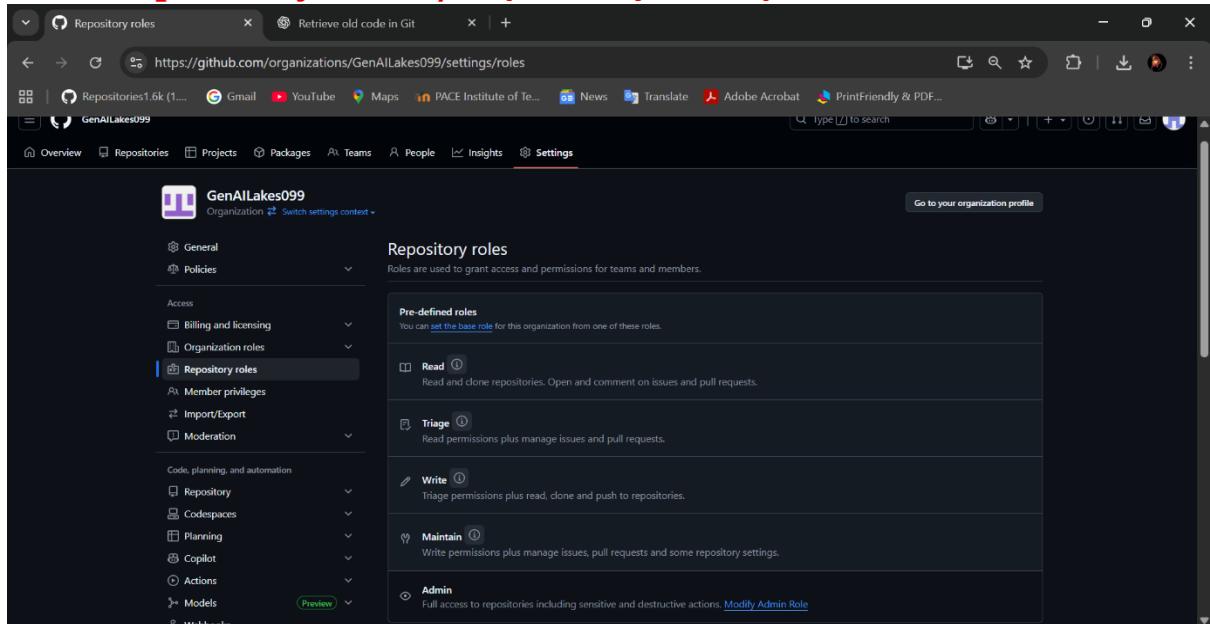
2. Role Assignment

→ Here you can assign specific role for users or team members

The screenshot shows the 'Assign teams or people an organization role' section of the GitHub organization settings. The sidebar on the left is identical to the previous screenshot. The main content area is titled 'Assign teams or people an organization role'. It features a search bar labeled 'Search organization role assignments', a dropdown menu set to 'Role: All', and a green button labeled 'New role assignment'. Below these are two tabs: 'Users' (0) and 'Teams' (0). A large empty box below the tabs displays the message 'No results'.



❖ Repository Roles/Repository Level permissions:



Collaborators & Team:

→ In the collaborators & team => manage access you can (**add peoples, create team, add teams**) by assigning the Permissions like (**Read, Write, Triage, Maintain, Admin**)

→ In the Add people invite the member by using the Email, GitHub Account, Phone Number) after you send invitation to the other side of the User can

accept or decline in there Github Account in the Notification.

→ After accepting the user, you can see in you **manage access** outside collaborators the previous pending notification will removed.

→ Now the User can also see your repository.

The screenshot shows the 'Collaborators and teams' section of a GitHub repository settings page. On the left, a sidebar lists 'Access', 'Collaborators and teams' (which is selected), 'Moderation options', 'Code and automation', 'Security', and 'Advanced Security'. The main area displays 'Public repository' status and three access categories: 'Base role' (Admin, 1 member), 'Direct access' (1 entity, 1 invitation), and 'Organization access' (0 users and 0 teams). A 'Manage' button is present under each category. Below this, the 'Manage access' section shows a list of users with their roles: 'Venkat@09' (Pending invite, Role: write). Navigation buttons for 'Previous' and 'Next' are at the bottom.

→ Another GitHub Account he want to Accept the Invitation

The screenshot shows a GitHub invitation notification. It features a profile picture of 'Venkat-Singareddy' and a plus sign icon. The text reads: 'Venkat-Singareddy invited you to collaborate on GenAllLakes099/Practice_to-add_Users'. Below this are two buttons: 'Accept invitation' (highlighted in green) and 'Decline invitation'. A note states: 'Owners of Practice_to-add_Users will be able to see: Your public profile information, Certain activity within this repository, Country of request origin, Your access level for this repository, Your IP address'. At the bottom, it asks 'Is this user sending spam or malicious content?' with options 'Block Venkat-Singareddy' and 'Report abuse'.

The image shows two screenshots of a GitHub repository named 'Practice_to-add_Users'.
The top screenshot displays the 'Access' section of the repository settings. It shows that the repository is a 'Public repository' with 'Admin' access level. Under 'Direct access', it indicates that 1 entity has access to the repository, specifically 'Venkat@09' with a role of 'Outside Collaborator'. There are buttons for 'Manage visibility', 'Manage', and 'Manage access'.
The bottom screenshot shows the main repository page for 'Practice_to-add_Users'. It lists 1 branch ('main') and 0 tags. A commit by 'Venkat-Singareddy' was made 15 hours ago, adding 'Colors'. The repository is public and has 0 forks, 0 stars, and 0 releases. It also has 0 packages published. A button to 'Add a README' is visible.

→ In the Collaborators you can (add peoples, create teams and add existing teams)

The screenshot shows the GitHub repository settings page for 'Practice_to-add_Users'. The 'Collaborators and teams' tab is selected. On the left, a sidebar menu includes 'General', 'Access' (selected), 'Moderation options', 'Code and automation', 'Copilot', 'Environments', 'Pages', 'Custom properties', and 'Security'. Under 'Access', 'Collaborators and teams' is also selected. The main area displays 'Public repository' status and three access categories: 'Base role' (Admin), 'Direct access' (0 members), and 'Organization access' (0 users and 0 teams). Below this is a 'Manage access' section with tabs for 'Direct access' (selected) and 'Organization access', and buttons for 'Create team', 'Add people', and 'Add teams'. A search bar and filters are also present.

User Permissions:

Read

- Recommended for non-code contributors who want to view or discuss your project.
- Read and clone repositories. Open and comment on issues and pull requests.

Triage

- Recommended for contributors who need to manage issues and pull requests without write access.
- Read permissions plus manage issues and pull requests.

Write

- Recommended for contributors who actively push to your project.
- Triage permissions plus read, clone and push to repositories.

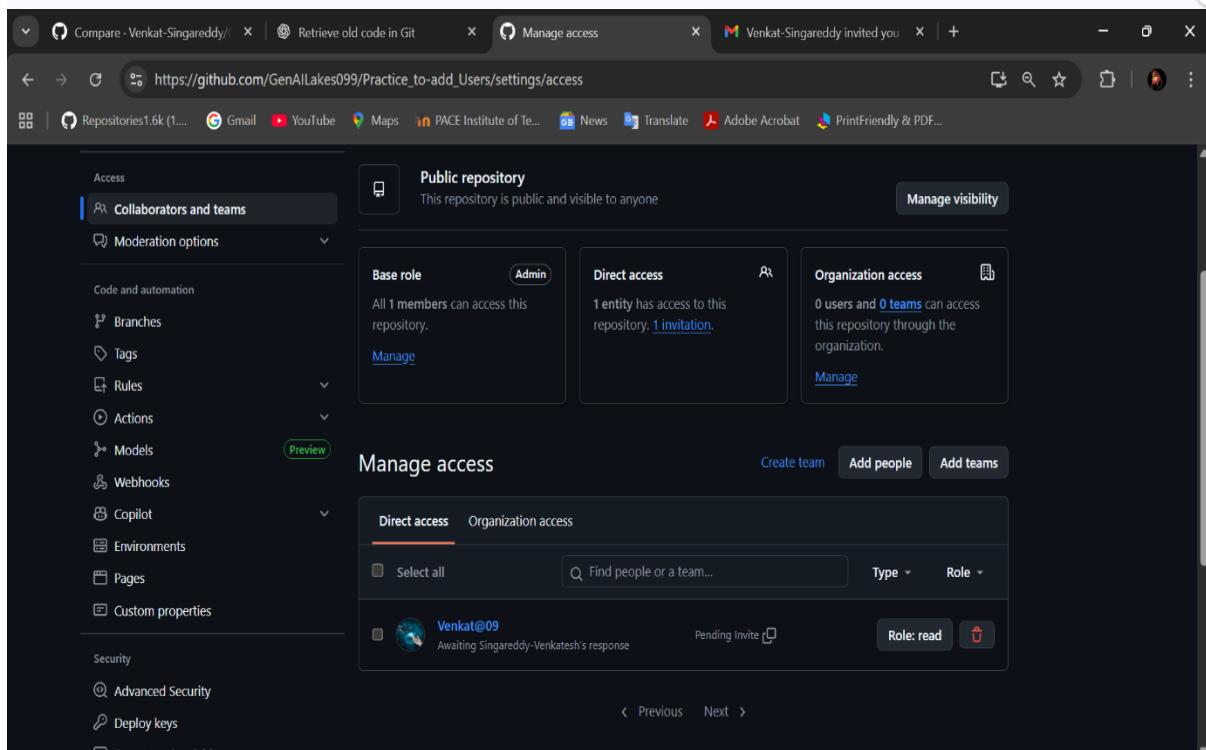
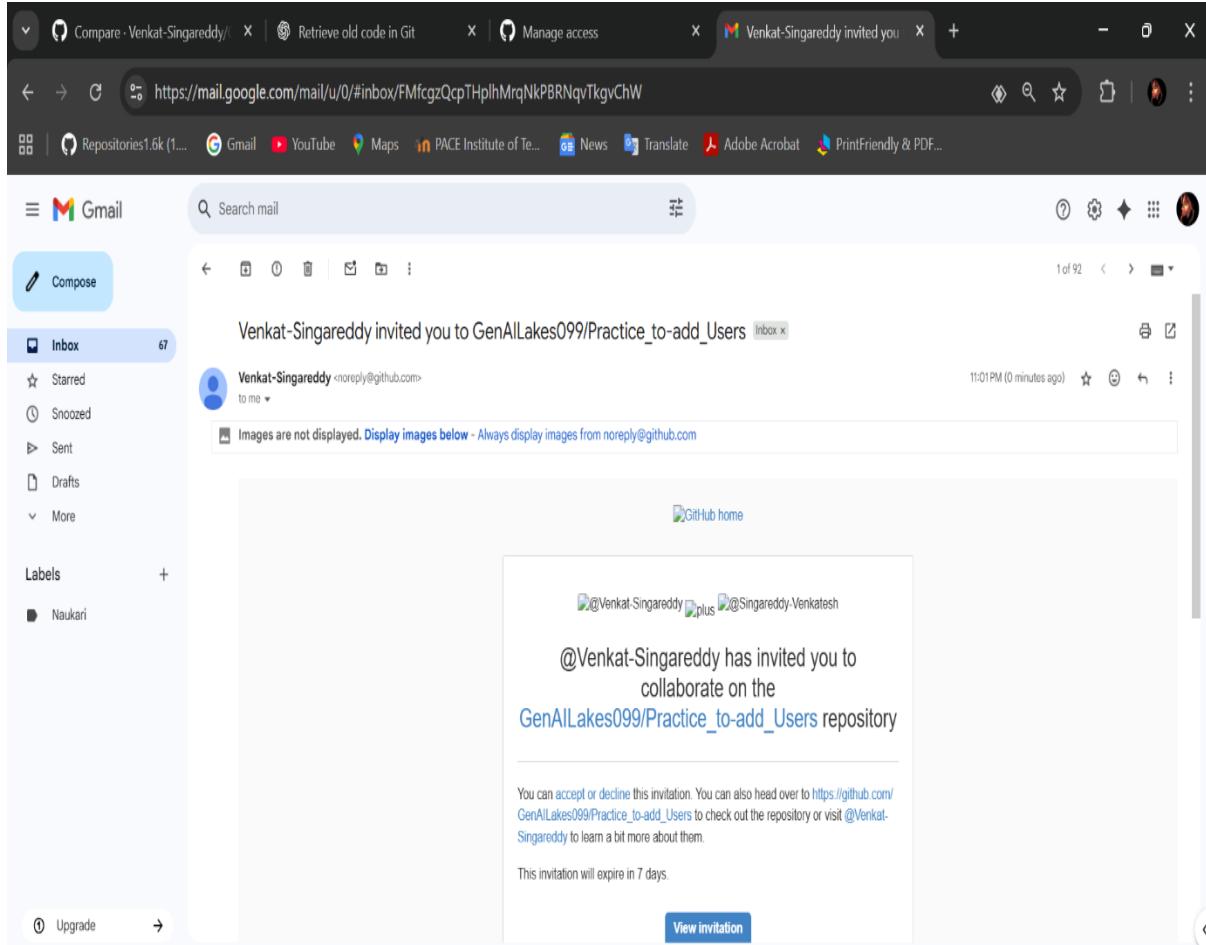
Maintain

- Recommended for project managers who need to manage the repository without access to sensitive or destructive actions.
- Write permissions plus manage issues, pull requests and some repository settings.

Admin (Base role)

- Recommended for people who need full access to the project, including sensitive and destructive actions like managing security or deleting a repository.
- Full access to repositories including sensitive and destructive actions.

→ Add members in your Repository Organization



→ You can also change the role or delete the Access

→ You Added Two Members in your Organization (Best recommend a minimum of two people within each organization have the owner role based on the team members)

The screenshot shows two browser tabs. The top tab is a Gmail inbox with a single email from GitHub (@noreply@github.com) inviting the user to join the @GenAllLakes099 organization. The email body contains a message from @Venkat-Singareddy, a note about the invitation expiring in 7 days, and a prominent green 'Join @GenAllLakes099' button. The bottom tab is the 'People' section of the GitHub organization settings for GenAllLakes099, showing two pending invitations listed under the 'Invitations' tab.

[GitHub] @Venkat-Singareddy has invited you to join the @GenAllLakes099 organization

@Venkat-Singareddy has invited you to join the @GenAllLakes099 organization

Hi Venkat@09!

@Venkat-Singareddy has invited you to join the @GenAllLakes099 organization on GitHub. Head over to <https://github.com/GenAllLakes099> to check out @GenAllLakes099's profile.

This invitation will expire in 7 days.

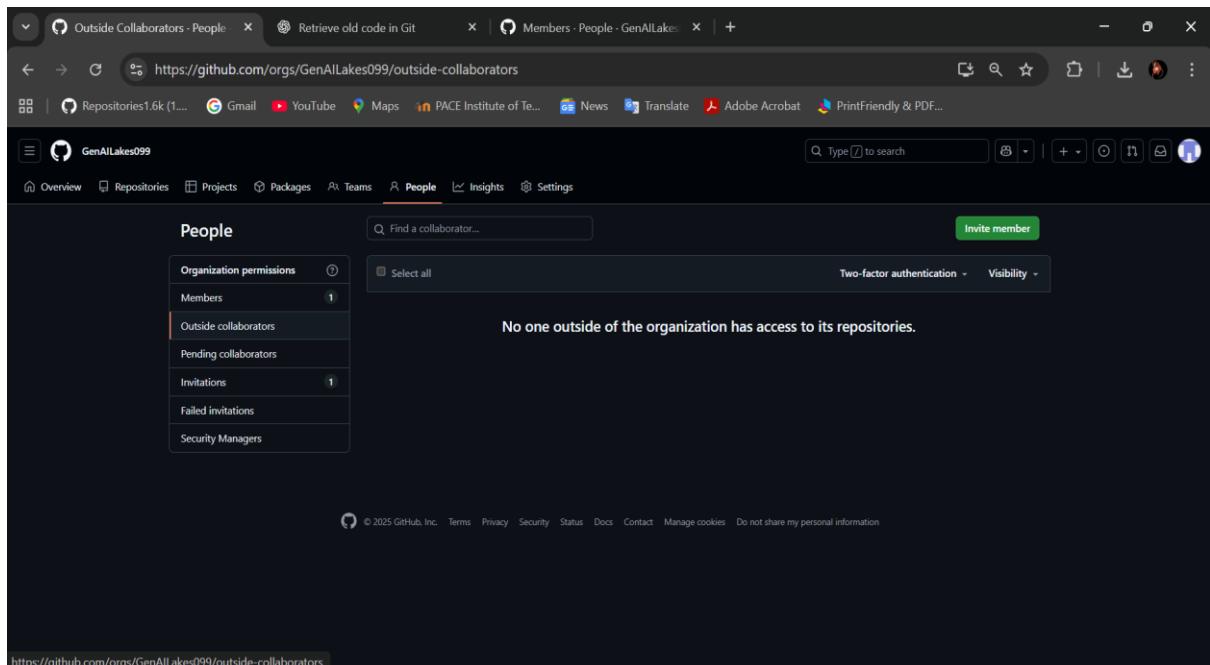
[Join @GenAllLakes099](#)

Note: If you get a 404 page, make sure you're signed in as Singareddy-Venkatesh. You can also accept the invitation by visiting the organization

2 invitations

Member	Invited on	by	...
Venkateswarlu44	Aug 25, 2025	Venkat-Singare...	...
Venkat@09 Singareddy-Venkatesh	Aug 25, 2025	Venkat-Singare...	...

→ **Outside Collaborators:** No one outside of the organization has access to its repositories we can restrict the outside users.



Branch Level Protection:

→ You can select and Protect your Repository Branches of your Organization

Branch rules:

- ❖ **Restrict creations:** Only allow users with bypass permission to create matching refs.
- ❖ **Restrict updates:** Only allow users with bypass permission to update matching refs.
- ❖ **Restrict deletions:** Only allow users with bypass permissions to delete matching refs.
- ❖ **Require linear history:** Prevent merge commits from being pushed to matching refs.
- ❖ **Require signed commits:** Commits pushed to matching refs must have verified signatures.
- ❖ **Require merge queue:** Merges must be performed via a merge queue.
- ❖ **Require deployments to succeed:** Choose which environments must be successfully deployed to before refs can be pushed into a ref that matches this rule.
- ❖ **Require a pull request before merging:** Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.
- ❖ **Require status checks to pass:** Choose which status checks must pass before the ref is updated. When enabled, commits must first be pushed to another ref where the checks pass.
- ❖ **Block force pushes:** Prevent users with push access from force pushing to refs.
- ❖ **Require workflows to pass before merging:** Require all changes made to a targeted branch to pass the specified workflows before they can be merged.
- ❖ **Require code scanning results:** Choose which tools must provide code scanning results before the reference is updated. When configured, code scanning must be enabled and have results for both the commit and the reference being updated.

Restrictions (Enterprise)

- ❖ **Restrict commit metadata:** Restrict commit author email addresses, committer email addresses, commit message content, and other metadata
- ❖ **Restrict branch names:** Restrict branch names

Branch Protection rule:

Note: Define branch rules to disable force pushing, prevent branches from being deleted, or require pull requests before merging.

- ❖ Add Branch Ruleset
- ❖ Add Classic Branch Protection Rule

→ In the branch ruleset in the “Ruleset name” suppose if you have multiple feature branches you have to give permissions to all the branches at a time simple type (feature*) so the permissions applicable to all the feature branches.

The screenshot shows two overlapping GitHub interface windows. The top window is titled 'Branch protection rules' and shows a single rule for the 'main' branch. The bottom window is a 'Propose changes' dialog for a commit to the 'main' branch, which is highlighted as a protected branch. The commit message field contains 'Update Users'. The dialog also includes fields for 'Extended description' and a note about the main branch being protected. It offers options to 'Create a new branch' or 'Propose changes'.

Note: In the above image it restricted to commit the code in the **main branch**.

→ In the above scenario when you created a ruleset to Main/Master branch it is very important to protection in the production level, so when the collaborators of your organizations are tried to merge his code in the main branch it will be restrict.

→ Without an approval from your **Admin**, you can't make changes in the main branch so it is highly protected.

→ when the user makes pull request, so first the **Admin Access members** they can check and review the code before merging in the production environment here we can protect our production code.

The screenshot shows a GitHub pull request page for a repository named "Practice_to-add_Users". The pull request has the title "added black color #1" and is currently in a "Review required" state. A comment from "Singareddy-Venkatesh" states: "I was added another color that is Black". Below the comment, there are two notifications: "Review required" (At least 1 approving review is required by reviewers with write access) and "Merging is blocked" (At least 1 approving review is required by reviewers with write access). The pull request has 1 commit and 1 file changed. On the right side, there are sections for Reviewers, Assignees, Labels, Projects, Milestone, and Development. The "Development" section notes that successfully merging the pull request may close certain issues. At the bottom, there is a "Finish your review" dialog box with a "Great Job!" message and options to Comment, Approve, or Request changes. The "Approve" option is selected. The GitHub interface includes a header with tabs like Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings, along with various search and filter tools.

The image consists of three vertically stacked screenshots of a GitHub interface, showing the progression of a pull request.

Screenshot 1: Pull Request Pending Review

A pull request titled "added black color #1" is shown. A comment from "Venkat-Singareddy" asks, "why are you added this colour in this file". Below the comment, there is a "Reply..." input field and a "Resolve conversation" button. A green box highlights the "Changes approved" status, indicating one approving review by a reviewer with write access. The status also shows "No conflicts with base branch". A "Merge pull request" button is present at the bottom.

Screenshot 2: Pull Request Merged

The pull request has been merged. A message states, "Venkat-Singareddy merged 1 commit into main from Singareddy-Venkatesh-patch-1 now". A purple box highlights the "Pull request successfully merged and closed" message. A "Delete branch" button is visible. An "Add a comment" section is at the bottom.

Screenshot 3: Repository Code View

The repository "Practice_to-add_Users" is viewed. The "Users" file is selected in the sidebar. The code content shows:

```
1 White
2 Orange
3 Green
4 Blue
5 Black
```

Member Privileges

→ You can give the specific members Privileges like User permissions in your organization

❖ 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.

❖ Repository creation

Members will be able to create only selected repository types. Outside collaborators can never create repositories.

Public: Members will be able to create public repositories, visible to anyone.

Private: Members will be able to create private repositories, visible to organization members with permission.

❖ Repository forking

Allow forking of private repositories:

If enabled, forking is allowed on private and public repositories. If disabled, forking is only allowed on public repositories. This setting is also configurable per-repository.

❖ Repository discussions

Allow users with read access to create discussions:

If enabled, all users with read access can create and comment on discussions in GenAILakes099's repositories.

If disabled, discussion creation is limited to users with at least triage permission.

Users with read access can still comment on discussions.

➤ Projects base permissions

Projects created by members will default to the selected role below.

❖ Pages creation

Members will be able to publish sites with only the selected access controls.

Public: Members will be able to create public sites, visible to anyone.

Private: Members will be able to create private sites, visible to anyone with permission.

❖ Integration access requests

Allow integration requests from outside collaborators:

Outside collaborators will be able to request access for GitHub or OAuth apps to access this organization and its resources.

➤ Admin repository permissions

❖ Repository visibility change

Allow members to change repository visibilities for this organization:

If enabled, members with admin permissions for the repository will be able to change its visibility. If disabled, only organization owners can change repository visibilities.

❖ Repository deletion and transfer

Allow members to delete or transfer repositories for this organization:

If enabled, members with admin permissions for the repository will be able to delete or transfer **public** and **private** repositories. If disabled, only organization owners can delete or transfer repositories.

❖ Issue deletion

Allow repository administrators to delete issues for this organization:

If enabled, members with admin permissions for the repository will be able to delete issues. If disabled, only organization owners can delete issues.

➤ Member team permissions

❖ Team creation rules

Allow members to create teams:

If enabled, any member of the organization will be able to create new teams. If disabled, only organization owners can create new teams.

The screenshot displays two stacked screenshots of the GitHub Organization settings interface.

Top Screenshot: Member privileges

- Left Sidebar:** Shows navigation categories like General, Policies, Access, Code, planning, and automation, Security, and Secrets & variables.
- Right Content Area:** Under "Member privileges", it shows "Set everyone's base permissions for your code".
 - Base permissions:** Describes how base permissions apply to all members and excludes outside collaborators. It includes sections for **No permission**, **Read**, **Write**, and **Admin**.
 - Organization member permissions:** Details what members can do based on their role.
 - Repository permissions:** Details what members can do at the repository level.

Bottom Screenshot: Secrets & variables

- Left Sidebar:** Shows navigation categories like Third-party Access, Integrations, and Developer settings.
- Right Content Area:** Shows "Secrets & variables" settings.
 - Projects base permissions:** Set to "Write".
 - Pages creation:** Set to "Public".
 - Integration access requests:** Set to "Allow integration requests from website collaborators".
 - Admin repository permissions:** Set to "Allow members to change repository visibility for this organization".
 - Repository deletion and transfer:** Set to "Allow members to delete or transfer repositories for this organization".
 - Issue deletion:** Set to "Allow repository administrators to delete issues for this organization".

➤ Moderation

❖ Block Users

The screenshot shows the GitHub Organization Settings page for 'GenAllLakes099'. The 'Settings' tab is selected. On the left, the 'Moderation' section is expanded, showing 'Blocked users' as the active sub-section. The main content area is titled 'Block a user' and explains that blocking a user prevents interactions across all repositories. It lists several actions that are blocked: opening or commenting on issues/pull requests, starring, forking, or watching repositories, and adding or editing wiki pages. A search bar allows searching by username, full name, or email address. Below the search bar are 'Block options' and a large red 'Block user' button. A message at the bottom states 'You have not blocked any users.'

❖ Interaction Limits

- ❖ Temporarily restrict which external users can interact with your repositories (comment, open issues, or create pull requests) for a configurable period of time. Users who are members of this organization will not be affected by these limits.
- ❖ This may be used to force a "cool-down" period during heated discussions or prevent unwanted interactions.
- ❖ Interaction limits may already exist in your organization's **public repositories**. Any changes here will override those limits.

The screenshot shows the GitHub Organization Settings page for 'GenAllLakes099'. The 'Settings' tab is selected. On the left, the 'Moderation' section is expanded, showing 'Interaction limits' as the active sub-section. The main content area is titled 'Temporary interaction limits' and explains that it restricts external users from interacting with repositories for a configurable period. It notes that members of the organization are not affected. A message states 'This may be used to force a "cool-down" period during heated discussions or prevent unwanted interactions.' Below this, it says 'Interaction limits may already exist in your organization's **public repositories**. Any changes here will override those limits.' There are three sections for limiting interactions:

- Limit to existing users:** Restricts users who have recently created their account. It includes categories: New users (red X), Users (green checkmark), Contributors (green checkmark), Collaborators (green checkmark), and Organization members (green checkmark). An 'Enable' button is present.
- Limit to prior contributors:** Restricts users who have not previously committed to the default branch of a repository in the organization. It includes categories: New users (red X), Users (red X), Contributors (green checkmark), Collaborators (green checkmark), and Organization members (green checkmark). An 'Enable' button is present.
- Limit to repository collaborators:** Restricts users who are not collaborators of a repository in the organization. It includes categories: New users (red X), Users (red X), Contributors (red X), Collaborators (green checkmark), and Organization members (green checkmark). An 'Enable' button is present.

❖ Code Review Limits

The screenshot shows the GitHub organization settings page for 'GenAllLakes099'. The left sidebar has a 'Moderators' section selected. The main content area is titled 'Code review limits' and contains the following text:
Restrict users who are permitted to approve or request changes on pull requests in public repositories within this organization.
Code review limits may already be specified by individual repositories. Any changes here will override those limits until unset.
Code review limits are currently managed individually for all repositories. Enable limits to permit only users who have explicitly been granted access to each repository to submit reviews that "approve" or "request changes". Remove limits to allow all users to submit pull request reviews. All users able to submit comment pull request reviews will continue to be able to do so.

At the bottom of the content area are two buttons: 'Limit reviews on all repositories' and 'Remove review limits from all repositories'.

➤ Moderators

- ❖ You can add organization members or teams as moderators for your organization. Moderators can block and unblock users from the organization, minimize comments, and manage interaction limits for all public organization repositories.
- ❖ You may add up to **10** members or teams as moderators.

The screenshot shows the GitHub organization settings page for 'GenAllLakes099'. The left sidebar has a 'Moderators' section selected. The main content area is titled 'Moderators' and contains the following text:
You can add organization members or teams as moderators for your organization. Moderators can block and unblock users from the organization, minimize comments, and manage interaction limits for all public organization repositories.
You may add up to **10** members or teams as moderators.

Below this text is a search bar with the placeholder 'Add a member or team'. At the bottom of the content area is a message: 'You don't have any moderators for this organization.'

Uses of Moderation in GitHub Organization:

1. Block Users

- What it does: Blocks specific users across the entire organization, not just one repository.
- Why useful:
 - Stops spammers or abusive contributors from opening issues, PRs, or commenting.
 - Provides consistency (blocked user is blocked everywhere in the org).
 - Saves admins from repeatedly blocking the same person repo by repo.

Example: If someone spams multiple repos in your org, blocking them once at the org level removes them everywhere.

2. Interaction Limits

- What it does: Restricts who can create issues, discussions, or PRs.
Options include:
 - Everyone (default, open contribution)
 - Prior contributors only
 - Org members only
 - Members active for X days
- Why useful:
 - Prevents “drive-by” spam from new accounts.
 - Protects sensitive or enterprise projects.
 - Encourages higher-quality contributions.

Example: In an open-source project, you can allow only past contributors to open issues to reduce spam.

3. Code Review Limits

- What it does: Controls who can review pull requests within the organization.
 - Can restrict reviews to org members or specific teams.

- Ensures only trusted users can approve code changes.
- Why useful:
 - Improves security (prevents malicious outsider reviews).
 - Keeps compliance with internal policies.
 - Ensures code quality by allowing reviews only from authorized maintainers.

Example: For a company repo, you can make sure only the DevOps or Security team can approve changes to Dockerfile.

4. Moderators

- What it does: Assigns designated moderators (in addition to org admins).
 - Moderators can enforce rules, block/report users, and manage discussions/issues.
- Why useful:
 - Distributes responsibility (not just admins handle abuse).
 - Keeps community healthy in large open-source projects.
 - Provides quicker response to violations.

Example: In a big org, you can appoint trusted contributors as moderators to handle spam/abuse reports without giving them full admin rights.

Summary – Why Moderation is Useful for Your Org

- Block Users → Protect from repeat abusers' org-wide.
- Interaction Limits → Reduce spam & ensure only trusted contributors interact.
- Code Review Limits → Secure your review workflow, maintain code quality.
- Moderators → Share responsibility, keep your community safe and productive.

Without Moderation



- You must block a spammer repo by repo
- Anyone can open issues/PRs freely
- Anyone with access may review PRs
- Only org admins can handle abuse

With Moderation



VS

- One block applies org-wide
- Limit interactions to members or trusted users
- Only authorized teams can review & approve
- Trusted users can be delegated as moderators

comparison table that clearly shows the difference between running a GitHub Organization without moderation vs with moderation

Feature	∅ Without Moderation	✓ With Moderation
Block Users	You must block a spammer repo by repo. They can still attack other repos in the org.	One block applies org-wide, stopping abuse everywhere at once.
Interaction Limits	Anyone (including spam bots/new accounts) can open issues/PRs freely.	Limit interactions to members, prior contributors, or trusted users, reducing spam and noise.
Code Review Limits	Anyone with access may review PRs; risk of low-quality or malicious approvals.	Only authorized teams/org members can review & approve, ensuring secure and high-quality merges.
Moderators	Only org admins can handle abuse, leading to delays in large orgs.	Trusted users can be delegated as moderators, distributing responsibility and responding faster.

Feature	Without Moderation	With Moderation
Community Safety	Higher risk of harassment, spam, and irrelevant issues/PRs.	Safer, more professional community with clear moderation policies.
Scalability	Admins overloaded with manual blocking/reporting.	Moderation scales well for large orgs or open-source communities.

Takeaway

- Without Moderation → More open, but higher risk of spam, abuse, and security issues.
- With Moderation → Controlled, safer environment that scales well for both enterprise organizations and large open-source projects.

Two-factor authentication:

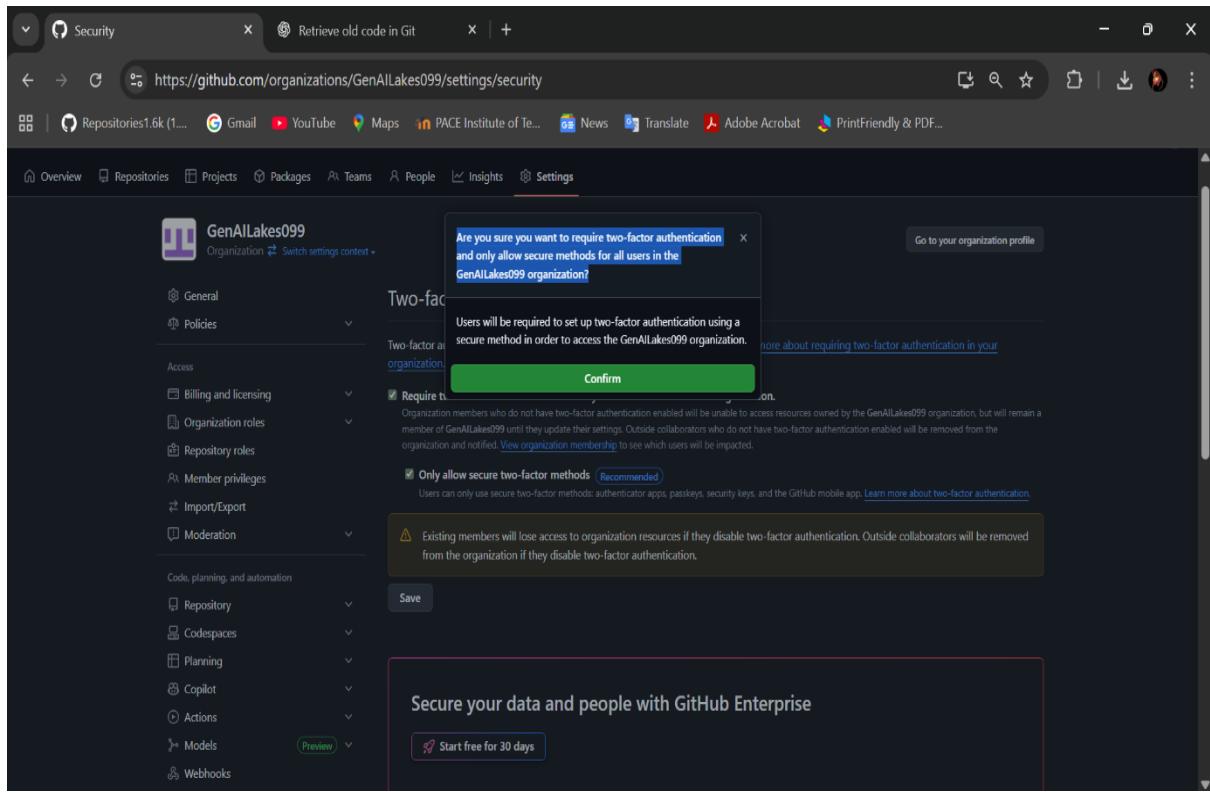
Go to the Organization → Security (Authentication Security) → Two-factor authentication (Email, SMS, Passkey, etc)

The screenshot shows the GitHub Organization settings interface for 'GenAllLakes099'. The left sidebar has 'Security' selected under 'Authentication security'. The main content area is titled 'Two-factor authentication' and contains two configuration options:

- Require two-factor authentication for everyone in the GenAllLakes099 organization.** Description: Organization members who do not have two-factor authentication enabled will be unable to access resources owned by the organization, but will remain a member of the organization and may update their settings. Outside collaborators who do not have two-factor authentication enabled will be removed from the organization and notified. [Learn more about organization members](#).
- Only allow secure two-factor methods (Recommended).** Description: Use one or only use secure two-factor methods: authenticator apps, passkeys, security keys, and the GitHub mobile app. [Learn more about two-factor authentication](#).

A note below states: "Existing members will lose access to organization resources if they disable two-factor authentication. Outside collaborators will be removed from the organization if they disable two-factor authentication." A 'Save' button is at the bottom of the form.

At the bottom of the page, there's a 'Secure your data and people with GitHub Enterprise' section with links to SAML single sign-on, Team Synchronization, SSH certificate authorities, IP allow list, and Certified for compliance.



POPUP: Are you sure you want to require two-factor authentication and only allow secure methods for all users in the GenAllLakes099 organization.