https://github.com/2DegreesInvesting/ds-incubator/issues/14

https://twitter.com/mauro_lepore

Access permissions for a GitHub organization

Situation

You want to propose changes to a source repo on a GitHub organization.

- Set the usual plumbing (`create_from_github()`).

- Initiate and work on your pull request (PR) (`pr_init()`).

- Synchronize, push, and submit your PR (`pr_sync()`, `pr_push()`).

Set the usual plumbing

- Fork source repo to your user account.

- Create a local clone of your fork.

- Connect your local clone with your fork (add remote origin).

- Connect your local clone with the source repo (add remote upstream).

Initiate and work on your PR

- Create a local branch to host PR.

- Change the local clone.

- Commit informing what changed and why.

Synchronize, push, and submit your PR

- Synchronize the source repository to get the latest changes.

- Push your PR to your fork.

- Submit your PR from your fork to the source repo.

But something goes wrong

The set up involves too many steps.

Unless you do it systematically (e.g. with usethis) sooner or later you will forget some step.

For example, these are my most common mistakes:

- I copy the wrong URL so remote origin points to the source repo instead of my fork.

- I forget to create a new branch and add commits to my local `master` branch.

What happens when you push?

1. Your commit(s) are now in the PR branch of your fork.

2. Your commit(s) are now in the master branch of your fork.

3. Your commit(s) are now in the PR branch of the source repo.

4. Your commit(s) are now in the master branch of the source repo.

5. You fail to push and get an error.

6. It depends on your role and permission level:

    · Roles: Owner, …, member, collaborator.
    · Permission levels: Admin, …, push (write), read.

Owner, billing manager & member

Organization members can have owner, billing manager, or member roles.

Owners have complete administrative access to your organization, while **billing managers** can manage billing settings. **Member** is the default role for everyone else.

Members, teams & outside collaborators

- You can set the default of what **members** can do for all repos.

- You can manage access permissions for multiple members at a time with **teams**.

- You can invite **outside collaborators** to give them access to **specific repositories**.

Permission levels for repositories

- **Read**: Recommended for non-code contributors who want to view or discuss your project.

- **Triage**: Recommended for contributors who need to proactively manage issues and PRs without write access.

- **Write**: Recommended for contributors who actively push to your project.

- **Maintain**: Recommended for project managers who need to manage the repository without access to sensitive or destructive actions.

…

Permission levels for repositories (continued)

…

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

Organization owners have admin permissions for every repository owned by the organization ([table](#)**).**

Example

Org settings: People

An owner's privileges

2° Investing Initiative

Repositories 52   Packages   **People 30**   Teams 5   Projects 4   Settings

2diiKlaus

Role: Owner ▾

Owner

As an owner, **2diiKlaus** has **admin access to all repositories** that belong to the 2DegreesInvesting organization. Manage your owners on the People page.

Org settings: Member privileges

2i 2° Investing Initiative

📱 Repositories **52**   📦 Packages   👤 People **30**   🎽 Teams **5**   📋 Projects **4**   ⚙ **Settings**

Organization settings
Profile
Member privileges
Billing
Security
Verified domains
Audit log
Webhooks
Third-party access
Installed GitHub Apps

Member repository permissions

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.

Read ▾

Organization member permissions

Admin
Members will be able to clone, pull, push, and add new collaborators to all repositories.

Write
Members will be able to clone, pull, and push all repositories.

ository types. Outside collaborators can never create repositories.

sible to anyone. Why is this option disabled?

Repo settings: Collaborators & teams

Repo settings: Branches

2DegreesInvesting / **r2dii.match** `Private`

👁 Unwatch ▾ 2   ★ Star 0   ⑂ Fork 2

<> Code   ⊘ Issues **1**   ⑁ Pull requests **0**   ▶ Actions   ▥ Projects **1**   ▤ Wiki   🛡 Security   �)⤷ Insights   ⚙ Settings

Options

Collaborators & teams

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Autolink references

Secrets

Actions

Default branch

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

master ▾   Update

Branch protection rules   Add rule

Define branch protection rules to disable force pushing, prevent branches from being deleted, and optionally require status checks before merging. New to branch protection rules? Learn more.

`master`   Currently applies to 1 branch   Edit   Delete

- People should have just the access they need.
- Be pragmatic.

"You do not rise to the level of your goals. You fall to the level of your systems" – James Clear, Atomic Habits