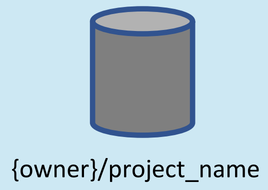
1. Organize your project on GitHub

1. Create a project repository on GitHub (https://www.github.com).
Ideally, use one of the templates to start.

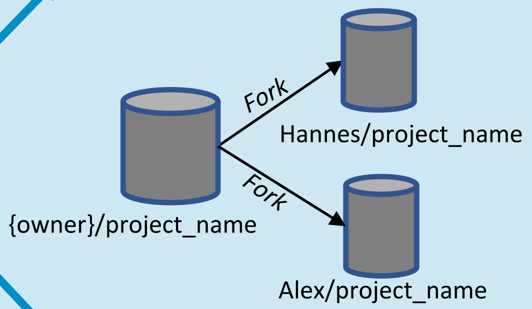


2. Next, decide on whether to work privately (e.g., for confidential projects), or publicly (e.g., for open-source projects)

We use GitHub "repositories" to collaborate in empirical research projects.

- **Code**: Find the source code written for your project (e.g., data cleaning, analysis)
- **Issues**: Keep a to-do list and assign items to team members
- **Projects**: Use the project board to define deadlines and organize your team meetings.

2a) Working on an open-source project
Team members fork (=copy) the repository to their own GitHub account.



2b) Working on a (privately shared) project
The repository owner adds team members as collaborators (settings → manage access → add people)

3a) Team members check out <u>forks</u> to their computers.
Go to a directory → right click → "Git Bash here". Finally, type cd {project_name}. Now, clone the repository!

```
git clone [your url]

git clone github.com/
Alex/{project_name}
```

3b) Team members check out the owner's repository to their computers.
Go to a directory → right click → "Git Bash here". Finally, type cd {project_name}. Now, clone the repository!

```
git clone [owner url]

git clone github.com/
owner/{project_name}
```

4. Follow the Git Workflow
(see below)

5a) Contribute changes from local repository back to the main repository
Sync your local fork with the owner's repository, using so-called pull requests.

See this link (TSH link) to find out how to perform pull requests!

5b) Since you work in the owner's repository, all changes are visible to all team members immediately.

2. The Git Workflow

1. Find an issue to work on (i.e., select one from the Issues or Project page)

If forked – check if fork is up-to-date using git fetch. If necessary, use git pull to retrieve the latest changes.

Whenever working on a project, we follow the Git workflow.

2. Create a branch
Recommended when working with others to avoid conflicts when pulling and pushing changes.

- git branch {name} → *Creates new branch*
- git checkout {name} → *Switches to this branch*
- In the future, use git pull {name} to get latest changes from this branch

3. Start working on the issue
Make the necessary changes in the source code of the files in the repository. Committing your changes can be done through Git Bash or by using an editor such as RStudio.

1. Git Bash	2. RStudio
• git status • git add {file name that was changed} • git commit –m [message of what you did] • git push (if first time: git push –u origin {branch name}	• In the top right, click on "Git" • Select files with changes you want to commit and click on "commit" • Enter a brief commit message and click "commit" • Finally, click on "push" to push your changes

The Git push command uploads the contents of your local repository to a remote repository on Github!

4. Update the Git "Issue" by letting others know what you changed/requesting feedback

Finally, update the issue, if necessary, with something that still needs to be done or close the issue if everything was done successfully. Tips on how to write good issues can be found on the TSH page (tilburgsciencehub.com/write/issues)!

If you worked on a forked repository, follow step 5A above to contribute your changes to the main repository.