Version control using Git and GitHub

Git is a source control management tool that tracks changes to files over time and allows users to manage and document the evolution of a project. GitHub is a web-based platform that hosts Git repositories and provides additional tools for collaboration, automation, documentation, and issue tracking.
While this protocol cannot be exhaustive, it introduces the most basic operations such as setting up a repository, recording file updates through commits, working with branches and forks, and merging contributions into the main version via pull requests. It also provides recommendations on commit practices, file embedding, privacy, and project sharing that will benefit research groups who want to adopt Git and GitHub not only for code, but also for notebooks, documentation, protocols, and other project deliverables.

Reviewed: May 12, 2026

This is a bench card. Full protocol available online.

Procedures

⊞ Optional: *Creating a GitHub account*

(1.) Create a GitHub account at https://github.com/.

›› *Working with Git on a local machine*

(1.) Confirm Git is installed by running `git --version` in the terminal. If not, install Git using your system's package manager or download it from https://git-scm.com/downloads.

(2.) *Optional:* Execute the following commands to associate commits with your GitHub identity.

```
git config --global init.defaultBranch main
git config --global user.name my-gh-name
git config --global user.email id+my-gh-name@users.noreply.github.com
```

(3.) Navigate to the project folder, then initialize a new Git repository with

```
cd /path/to/my-local-repository
git init
```

(4.) Create or modify files in the folder.

(5.) Inspect which files have changed.

```
git status
git diff
```

Add (“stage”) files or part of a file to include in the next snapshot:

```
git add .
git add my-file
git add -p my-file
```

(6.) *Optional:* To prevent certain files from being tracked, create a `.gitignore` file in the repository root and list patterns for files or directories that should be excluded on individual lines.

(7.) Create a versioned snapshot of the staged files.

```
git commit -m "Commit message"
```

(8.) Repeat editing, staging, and committing as needed. Use `git log` to view the commit history.

Advanced Git operations

(1.) *Optional:* Use `git stash` to temporarily save all tracked changes (both staged and unstaged) without committing. You can recover the most recent stash later using `git stash pop`.

(2.) *Optional:* To recover a previous state *after committing,* execute:

- `git reset --soft HEAD~1` to undo the last commit without deleting changes.
- `git reset --mixed HEAD~1` to undo the last commit and unstage the changes.
- `git reset --hard HEAD~1` to discard the last commit and all its changes completely.

> ***Critical:*** Any local changes that were part of that commit will be permanently discarded and cannot be restored with Git.

(3.) *Optional:* To undo changes in the working directory *before committing,* run

- `git reset my-file` to remove the file from the staging area and add further edits.
- `git checkout -- my-file` to discard all unsaved changes in the file.

> ***Critical:*** Only use this if you are sure you want to go back to the last committed version of that file.

(4.) *Optional:* At important milestones such as manuscript submissions, dataset releases, or publication, add semantic versioning such as `git tag v1.0.0`.

›› *Working with branches and merging changes*

(1.) View available branches with `git branch`. The current branch is marked with an asterisk.

(2.) Create a new branch and/or switch to it.

```
git branch my-feature
git checkout my-feature
```

(3.) Edit, stage, and commit changes on the new branch as usual.

(4.) Update your new branch with the latest changes from the main branch.

```
git fetch origin
git merge origin/main
```

(5.) When done, switch back to the main branch. After you merge your feature branch, you can delete it.

```
git checkout main
git merge my-feature
git branch -d my-feature
```

›› *Working with remote repositories on GitHub*

(1.) Create a new repository on GitHub using the web interface.

- Make the repository *private*. You can publish the repository later.
- Initialize the repository with a `README.md` and a `.gitignore` file.

> ***Critical:*** Before first push, confirm that `.gitignore` excludes any files that contain private data, credentials, personal identifiers, or unpublished results. GitHub is public by default and search engines can index it quickly.

- If institutional guidelines do not otherwise specify, choose a permissive license for your work: 🕮

License	Attribution	Share alike	Commercial use	Patent clause	Recommended use
CC0-1.0	No	No	Yes	No	Public domain datasets; templates; code snippets
CC-BY-4.0	Yes	No	Yes	No	Protocols, educational content, documentation
MIT	Yes	No	Yes	No	Code for figures, small analysis scripts
Apache 2.0	Yes	No	Yes	Yes	Reusable libraries, substantial academic/industry software
GPLv3	Yes	Yes	Yes	Yes	Full analysis pipelines where openness of all modifications is required

  Please do not add a license to the repository if you are unsure. Software and documentation licenses are not revocable once published and may affect collaborators and reuse policies.

(2.) *Critical:* Generate a fine-grained personal access token (PAT) for remote authentication:

- Go to https://github.com/settings/personal-access-tokens/ and select “Generate new token”.
- Provide a token name and expiration date.
- Choose “All repositories” to allow access to future repositories, or explicitly select existing ones. 🕮
- Under “Repository permissions”, set “Contents” to “Read and write”.
- Click “Generate token” and copy the generated string to your clipboard. 🕮

(3.) Create a copy of the remote repository on your local machine, or link a local repository to the remote, by executing one of the following commands:

```
git clone gh-repo-url.git
git remote add origin gh-repo-url.git
```

(4.) To retrieve changes from the remote repository, execute: 🕮

```
git fetch origin
git merge origin/main
```

(5.) Upload your committed local changes to the remote repository with

```
git push origin main
```

›› *Collaborating on remote repositories with GitHub*

(1.) As repository owner, add collaborators to your repository; set privileges as appropriate. 🕮

(2.) *Optional:* As repository owner, create a development branch for the project once

```
git checkout -b dev
git push origin dev
```

(3.) As a collaborator, accept the invitation, pull the repository, and create a new branch for your edits to isolate changes and make reviewing easier.

```
git pull origin dev  # or "main" if no development branch exists
git checkout -b my-feature
```

```
git push origin my-feature
```

(4.) Edit, stage, commit, and push changes on your branch as usual.

> ***Critical:*** Pull the latest changes regularly from the remote repository to keep your local repository up to date.

(5.) Once your branch is ready to be merged, create a pull request on GitHub:

- Navigate to the repository on GitHub, click “Compare & pull request”.
- Respond to comments and update the pull request as needed.
- Once approved, the pull request will be merged into the development or the main branch.

›› *Linking repositories with submodules*

(1.) Use a submodule when part of one repository (private content, shared utilities, a vendored dependency) needs its own version history but should appear as a subdirectory of the parent. The parent pins a specific commit of the submodule, so updates are explicit and reproducible.

(2.) Add a submodule to the parent repository.

```
git submodule add gh-other-repo-url.git submodule/path/in/parent
git commit -m "Add submodule/path/in/parent as submodule"
```

(3.) When others clone the parent, initialize submodules in the same step or after the fact:

```
git clone --recurse-submodules gh-parent-repo-url.git
git submodule update --init --recursive
```

(4.) To update the submodule to a newer commit, update the pointer in the parent.

```
cd submodule/path/in/parent
git pull origin main
cd ..
git add submodule/path/in/parent
git commit -m "Bump submodule pointer"
```

> ***Critical:*** Editing files inside the submodule does *not* automatically update the parent's pointer. You must commit twice: once inside the submodule, once in the parent.

(5.) *Optional:* To remove a submodule:

```
git rm path/in/parent
git commit -m "Remove submodule path/in/parent"
```

›› *Preserving file history when renaming or splitting repositories*

(1.) For renames *within* a single repository, use `git mv` rather than a plain shell `mv` followed by `git add`.

(2.) Confirm Git detected the rename.

```
git status              # "renamed: old -> new"
git log --follow new    # history across the rename
```

(3.) To move files *between* repositories while preserving history, `git mv` is not enough — it only operates within one repository. Use `git filter-repo` to extract the relevant history into a portable form, then merge it into the target repository.

T › ***Repository transfer***

(1.) Audit the repository.

- No pending branches, unmerged pull requests, or unpublished changes?
- No dependencies on private data or credentials?

(2.) Finalize `README.md` to reflect the latest project title, description, and status of the project. 🕮

(3.) Create a final release tag so there is a citable, frozen snapshot of the code at time of handoff.

```
git tag v1.0.0-final
git push origin v1.0.0-final
```

(4.) *Optional:* For published work, link the repository to Zenodo. 🕮

(5.) *Critical:* If the repository is associated with a personal GitHub account, transfer the repository to a designated maintainer, the lab's GitHub account or the institutional GitHub organization. Ensure a maintainer is assigned for substantial software projects. 🕮

(6.) Remove the departing member's write/admin access after transfer is complete.

(7.) Archive the repository if it is no longer maintained. Archived repositories are read-only.

(8.) Add the repository title, archive status, and DOI to the lab's knowledgebase or data inventory.

List of references

🕮 Notes (available online)