**Adding a local repository to GitHub with GitHub CLI**

1. In the command line, navigate to the root directory of your project.
2. Initialize the local directory as a Git repository.

git init -b main

1. Stage and commit all the files in your project.

git add . && git commit -m "initial commit"

1. To create a repository for your project on GitHub, use the gh repo create subcommand. When prompted, select **Push an existing local repository to GitHub** and enter the desired name for your repository. If you want your project to belong to an organization instead of your user account, specify the organization name and project name with organization-name/project-name.
2. Follow the interactive prompts. To add the remote and push the repository, confirm yes when asked to add the remote and push the commits to the current branch.
3. Alternatively, to skip all the prompts, supply the path to the repository with the --source flag and pass a visibility flag (--public, --private, or --internal). For example, gh repo create --source=. --public. Specify a remote with the --remote flag. To push your commits, pass the --push flag. For more information about possible arguments, see the [GitHub CLI manual](https://cli.github.com/manual/gh_repo_create).

**To commit only certain and not all files**

**1. Get a list of files you want to commit**

$ git status

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: file1

modified: file2

modified: file3

modified: file4

**2. Add the files to staging**

$ git add file1 file2

**3. Check to see what you are committing**

$ git status

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: file1

modified: file2

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: file3

modified: file4

**4. Commit the files with a commit message**

$ git commit -m "Fixed files 1 and 2"

**5. If you accidentally commit the wrong files**

$ git reset --soft HEAD~1

**6. If you want to unstage the files and start over**

$ git reset

Unstaged changes after reset:

M file1

M file2

M file3

M file4















