GIT HUB HELPER

SUBMITTING WITH GIT

SUMMER COURSE

LINNAEUS
UNIVERSITY
SUMMER 2024

USING GIT & CLONE A REPO

Here’s a step-by-step guide on how to clone a GitHub repository and commit changes to that repo using Git Bash.

Use cd to navigate between directories.
**```cd ..```** moves you up one directory level.

1. One your computer, create a folder.

2. Make sure that the files you want to commit to the repo are in the appropriate folder you are aiming to commit.

3. To clone your repository to your local machine, you'll need the repository's URL.

   Go to your GitHub account and navigate to the repository you want to clone. Click on the green Code button.

   Copy the URL (it might look like: **https://github.com/username/repo.git).**

4. Open Git Bash

   In Git Bash, navigate to the directory where you want to store your project - where you want your cloned repo to “live” on your computer.

   For example:
   **```cd my/path/to/introduction_to_computerScience```**

   When you have navigated to **introduction_to_computerScience**, run the following command
   (Replace **https://github.com/username/repo.git** with the actual URL of your repository.):

   **```git clone https://github.com/username/repo.git```**

5 Once the repository is cloned, navigate into the directory of your repository (Replace repo with the name of your repository.):
**```cd repo```**

Navigate to the subdirectory (the folder for the specific week) within your repository where you want to add your new folder.

Use cd to change directories if needed. Navigate into the subdirectory for the appropriate week.
For example, if you want to add the folder to subdirectory:
**```cd subdirectory```**

6 Open your file explorer and drag and drop the folder into the target subdirectory of your local Git repository. You can use your file manager to move the folder into the repository subdirectory.

7 Check that the folder has been successfully added to the subdirectory.

In Git Bash, you can use **```ls```** to list the contents of the subdirectory. You should see your new folder listed.

8 The root of your Git repository is the top-level directory where the .git folder resides. This is the starting point of your repository. If you are already in the root of your repository, you can stage changes from there directly. Navigate to the root of your Git Repository.

Use **```cd ..```** to go up one level in directory hierarchy

Git tracks changes in your repository, but it doesn't automatically know which changes you want to commit. You have to stage the changes first. Use Git to stage the new folder and its contents so that Git tracks the changes.

If you want to stage all changes in the repository (including new folders and files), you can use:

**```git add .```**

or you can use the command

**```git add subdirectory/new-folder```**

8 Commit Your Changes - after staging your changes, the next step is to commit them. A commit is like a snapshot of your repository at a certain point in time.

The **-m** flag allows you to add a commit message directly from the command line. The message should briefly describe the changes you've made.
**```git commit -m "Add new-folder to subdirectory"```**

9 Finally, you'll want to push your changes to GitHub so that they're saved remotely and visible on your GitHub repository.

**```git push```**

If this is the first time you're pushing to this repository, Git may ask you for your GitHub username and password (or a token if you’ve set up two-factor authentication).

To make sure your changes were successfully pushed, go back to your GitHub repository in your browser and refresh the page. You should see your new folder and any files inside it.

ADD FILES/FOLDERS TO ALREADY CLONED REPO

If you already have a Git repository on your computer, meaning you've either cloned it before, you can manage and update it without needing to clone it again. Here's how you can work with your existing repository, including adding folders, staging changes, committing, and pushing updates.

1. Open Git Bash - start by opening Git Bash on your computer. Use the cd command to navigate to your existing repository directory. For example:

   ```
   cd path/to/your/repo
   ```

   Replace **path/to/your/repo** with the actual path to your repository.

2. Add a new folder to your repository - if you want to add a new folder or files, you can do so using your file explorer or directly from Git Bash.
   Using File Explorer: Open your file explorer, and drag and drop the folder into your repository directory. (You can also create folders directly in Git Bash, or move existing files to your repo in Git Bash, but we won't cover that here.)

3. Check if the folder has been successfully added by listing the contents. You should see your new folder listed.

   ```
   ls
   ```

4. Stage the changes - to stage the new folder and its contents, use git add. For example, if the new folder is named **introduction_to_computerScience** use the command

   ```
   git add introduction_to_computerScience
   ```

   f you want to stage all changes (including modifications, deletions, and new files), use:

   ```
   git add .
   ```

   This command stages all changes in the repository.

5 Commit Your Changes - after staging, commit your changes with a descriptive message. For example:

**```git commit -m "Added introduction_to_computerScience with its contents"```**

6 If the GitHub repository already contains commits (e.g., if it was initialized with a README or other files), run this command to ensure your local repository is up-to-date before pushing your changes. This step helps avoid conflicts with any existing commits on the remote repository. Use:

**```git pull```**

7 Push to Remote Repository - to push your committed changes to a remote repository (like GitHub), use:

**```git push```**

INITIALIZE A LOCAL GIT REPOSITORY AND PUSH TO GITHUB

Initializing a local Git Repository and pushing to GitHub involves setting up a version control system for your project on your local machine and then making it available on GitHub for backup. When you initialize a local Git repository, you create a .git directory in your local project folder. This directory contains all the metadata and configuration files that Git uses to track changes in your project.

You will create a new repository on GitHub first.
Go to the GitHub website and sign in to your account. Click on the "+" sign at the top right corner and select "New repository". Provide a name for your repository, choose the desired settings, and create the repository.

On the GitHub repository page, note the repository's URL (ends with .git). You will need it later in the guide.

1. Start by opening Git Bash.
   Navigate to Your Project Directory: Use the cd command to move to your project directory where you want to initialize Git - that will be your root folder for Git.

   **```cd path/to/your/project```**

2. Initialize a new Git repository in your project directory. This command creates a new .git directory in your project, which will track your changes:

   **```git init```**

3. Stage all your files for the first commit - this stages all files in your project directory:

   **```git add .```**

4. Make the first commit with a message:

   **```git commit -m "Initial commit"```**

5 You need to link your local repository to the existing GitHub repository. Use the URL of your GitHub repository to add a remote named origin.

Replace **https://github.com/username/repo.git** with the **URL of your newly created GitHub repository.**
You can find this URL on your GitHub repository page, typically under the "Code" button. Then run the command:

**```git remote add origin https://github.com/username/repo.git```**

6 Verify the remote repository - check that the remote has been added correctly. This will list the remote repositories linked to your local repository:

**```git remote -v```**

7 Push your local commits to the GitHub repository. By default, you push to the main branch (or master, if that’s the default branch name).
If your default branch is master, replace main with master.
Handle authentication - you might be prompted for your GitHub username and password.

Use:

**```git push -u origin main```**