**Git and GitHub cheat sheet**

\*Using git bash terminal on windows

1. Install GitHub CLI (called gh) via this [link](mailto:https://cli.github.com/).
2. Find the directory of the gh file.
3. Create a variable to save the gh file directory:
   1. cd "the path of the file" // "name" is for terminal to understand special characters and spaces
   2. ghFolder="$(pwd)/" // the command copy the current file directory im at. If I want to save a subfolder inside my current directory, write: ghFolder="$(pwd)/'subfolder directory'/"
   3. export PATH=$PATH:ghFolder // PATH is a os saved cpmmand for the local bin. This command moves the ghFolder content into the local bin so the os can use gh (GitHub CLI)
   4. check if it went well: gh --version
4. Connect GitHub user to Git bash terminal:
   1. gh auth login: // if its not working use: winpty gh auth login
      1. Choose GitHub Host: GitHub.com
      2. Choose authentication method: HTTPS/SSH // I chose HTTPS
      3. Complete authentication: web browser
      4. Check for authentication: gh auth status.
5. Create a new repo from existing directory:
   1. Create a variable holding the wanted directory. // change directories to the variable one
   2. Create a .git file with: git init
   3. Upload the files inside the directory: git add .
   4. Commit the changes: git commit -m "Short text to explain the commit"
   5. Create a new repo: gh repo create name-of-the-repo --private --source=. // can write --public instead of --private to make the repo open sourced.
   6. Create a new branch: git branch -M name-of-branch // the main branch of the code
   7. Push files to the repo: git push -u origin name-of-branch

### 1. GH Login Operation

\*\*Objective\*\*: Authenticate to GitHub from Git Bash using the GitHub CLI (`gh`).

- \*\*Step 1\*\*: Install GitHub CLI (if not already installed). Visit [GitHub CLI] (https://cli.github.com/) for installation instructions.

- \*\*Step 2\*\*: Open Git Bash.

- \*\*Step 3\*\*: Type `gh auth login` and press Enter.

- \*\*Step 4\*\*: Follow the prompts to authenticate. You'll choose GitHub.com, then either login via a web browser or paste a token.

- \*\*Step 5\*\*: If choosing web browser login, a browser window will open. Follow the instructions to authenticate.

### 2. Create My First Repo

\*\*Objective\*\*: Initialize a local repository and create a corresponding repository on GitHub.

- \*\*Step 1\*\*: Navigate to the directory where you want to create your repository using `cd your\_directory`.

- \*\*Step 2\*\*: Initialize the local repository with `git init`.

- \*\*Step 3\*\*: Create a new file or edit existing files. Stage files with `git add filename` or `git add .` for all changes.

- \*\*Step 4\*\*: Commit your changes with `git commit -m "Your commit message"`.

- \*\*Step 5\*\*: Create a new repository on GitHub using `gh repo create repo-name`. Follow the prompts to configure your repo.

- \*\*Step 6\*\*: Push your local commits to the newly created GitHub repository using `git push -u origin main`.

### 3. Create Pull Request

\*\*Objective\*\*: Propose changes from one branch to another within a repository.

- \*\*Step 1\*\*: Ensure your local branch has all the intended changes committed.

- \*\*Step 2\*\*: Push your branch to GitHub if it's not already there with `git push origin your-branch-name`.

- \*\*Step 3\*\*: Use `gh pr create` to create a pull request. Follow the interactive prompts to specify the base branch, PR title, and description.

- \*\*Step 4\*\*: The GitHub CLI will provide a URL to the newly created pull request. You can share this URL with reviewers.

### 4. .git and .gitignore Files

\*\*Objective\*\*: Understand the role of `.git` directory and `.gitignore` file.

- \*\*`.git` Directory\*\*: This hidden directory is created in your project directory upon `git init`. It stores all of the necessary repository data—branches, commits, configurations, etc.

- \*\*Creating a `.gitignore` File\*\*: Create a `.gitignore` file in your project root to list files and directories that Git should ignore.

- Use `nano .gitignore` or another text editor to create and edit this file.

- Add patterns of files to ignore, such as `node\_modules/`, `.DS\_Store`, etc.

- Save the file and commit it with `git add .gitignore` and `git commit -m "Add .gitignore"`.

### 5. Push New Code to Secondary Branch

\*\*Objective\*\*: Safely push new code to a secondary branch for review or collaboration.

- \*\*Step 1\*\*: Create a new branch with `git branch secondary-branch-name`.

- \*\*Step 2\*\*: Switch to your new branch with `git checkout secondary-branch-name`.

- \*\*Step 3\*\*: Make your code changes, then stage and commit them.

- \*\*Step 4\*\*: Push your new branch to GitHub with `git push -u origin secondary-branch-name`.

### 6. Merge Branches

\*\*Objective\*\*: Merge changes from one branch (e.g., feature branch) into another (e.g., main).

- \*\*Step 1\*\*: Switch to the receiving branch (e.g., `main`) with `git checkout main`.

- \*\*Step 2\*\*: Ensure the receiving branch is up to date with `git pull`.

- \*\*Step 3\*\*: Merge your feature branch into the current branch with `git merge feature-branch-name`.

- \*\*Step 4\*\*: Resolve any merge conflicts, commit the changes, and then push to the remote repository.

### 7. Basic Commands

- `git init`: Initialize a new Git repository.

- `git clone [URL]`: Clone a remote repository.

- `git add [file]`: Stage changes for commit.

- `git commit -m "[message]"`: Commit staged changes.

- `git status`: Check the status of changes.

- `git push`: Push committed changes to a remote repository.

- `git pull`: Fetch and merge changes from the remote repository.

### 8. Advanced Commands

- `git branch [branch-name]`: Create a new branch.

- `git checkout [branch-name]`: Switch to a specific branch.

- `git merge [branch]`: Merge a branch into the current branch.

- `git rebase] branch]`: Reapply commits on top of another base tip.

- `git stash`: Temporarily shelves changes so you can work on a different branch.

- `git log`: Show commit logs.

- `git diff`: Show changes between commits, commit and working tree, etc.