­­­Use of GitHub for Assignment

|  |
| --- |
| 1. **Accept link** to git-hub repository from sender. 2. Open git-bash on windows or bash on mac. **Move to directory** where you would like to store the git repository. E.g. cd Desktop 3. **Clone the repository**    1. git clone https://github.com/minogud2/softEngAssignment4    2. If problems encountered, make sure the SSH key is established before cloning repository. 4. Once established, **cd into the new directory**    1. cd softEngAssignment4 5. **Check out the existing branches**    1. git branch    2. A list of four branches should appear(see tree)- If you still see the previous branches I set up (base and working tree), git remote prune origin. If that doesn’t work then forcibly delete them by doing the following: git branch -D base and git branch -D WorkingTree 6. **Log into your branch**    1. git checkout Darragh    2. Bash will then display your change. See below.      1. **Get up to date version of master repository**    1. git pull master origin    2. Do this only, if you want the most up to date version of the repository. Ensure that you have not worked on any documents in your repository before doing this. Otherwise, you will pull from the master and overwrite your version of the repository. 2. **Begin working.**     1. Create folder in repository and use this to develop your code. We will then discuss which files need to be pushed to the main skeleton folders.    2. git add .– adds the files and stations them for commit.    3. git commit -m “This is my first commit” - Ensure all commits are relative to what you are doing. Everyone shouldn’t be writing- version1 commit or initial commit. Instead- “Json file parsed into database. Committing database- version 1”.    4. git status - check that the files have been updated.    5. git push origin master- or git push Pushes your completed work to the master for merging. If the push doesn’t work. Complete the following:       1. git push -u origin master-the first time that you push that branch. You only need to do it once, and that sets up the association between your branch and the one at origin in the same way as git branch --set-upstream does. 3. **Master will merge changes.**    1. As git administrator for the project, I will handle the merges for the time being. The changes will be merged by doing:    2. git checkout master    3. git merge Darragh |

Other useful information

|  |
| --- |
| **Adding branches**   * git branch NewBranchName   **Deleting Branches**   * git branch -d NewBranchName- deletes if there are no merge conflicts. * git branch -D NewBranchName- forcibly deletes irrespective of merge issues.   **Undo commits**   * if you accidently delete your files when merging, as I have done while making this document. It’s possible to get back your info using the following: * git reset * git stash * git add . * git commit -m “undoing delete” * git push   **Merging upstream changes into your local repository**   * git pull <remote>- does a git fetch and a git merge all in one**.** Brings the local branch up to date with the remote branches/master. It automatically merges changes without reviewing them first. May run into conflicts if branches not closely managed.   **Tutorials:**  For all aspects of Git:  <https://www.atlassian.com/git/tutorials>  For undoing a delete:  <http://stackoverflow.com/questions/927358/how-to-undo-last-commits-in-git> |

Deductive Process