**Git Commands**

NB: all of these are to be run in the Terminal in Posit Workbench (PWB) - the command lines always start with a '$' - should automatically be there

If you understand the different types of terminals/have set your PWB terminal options to something non-default, make sure you're in a 'Bash' terminal to run these commands

Key

* *Italics* – actual git command text you need
* <text> – replace this with the relevant text (without “” unless specified)
* [text] – notes on the functionality, don’t type them in as part of the commands!

**Checking status of Git world**

* *git status*
* *git diff* [view all changes you've made]

**Branches**

* Switching to pre-existing branch:
  + *git checkout* <branch name>
* Create new branch and switch to it:
  + *git checkout -b* <branch name>
* When you do this, you'll need to do the following to make sure that the remote (repository hosted on Github/Gitea) knows that there's a new branch added to the repository:
  + *git push --set-upstream origin* <branch name>
* To delete branches from local and remote, do the following:
  + *git remote prune origin* [prunes remote]
  + *git branch -d* <branch\_name> [deletes local]

**Fetching most recent changes (commits) to branches from the remote version on GitHub/Gitea**

* This both fetches the changes (git fetch), and merges them into the branch currently checked out (git merge) - if you want to update other branches, you'll need to check them out first then run this
  + *git pull*
  + NOTE: you can only do this when you have staged all the changes in your project (see below)
* To get the changes made to the main branch into a feature branch, do the following steps in order, then deal with any merge conflicts if they occur
  + *git checkout main*
  + *git pull*
  + *git checkout* <feature\_branch\_name>
  + *git merge main*

**Staging code and making commits**

* To stage files (i.e. 'add them' to bundle of code included in a single commit)
  + *git add* <file name> [single files 1 by 1]
  + *git add .* [all files you have changed]
* To make a commit (i.e. solidify the changes into history)
  + *git commit -m* "<commit message>"

Uploading changes to the remote repository on GitHub/Gitea

* *git push*
* NOTE: often good idea to call `*git pull*` before doing this to make sure your branch is up to date

**Tagging major versions of code**

This should be done when a version of code has been used to produce an output i.e. KPI anlaysis dataset/output/publication. It can be done based on the current version of the ‘main’ branch, or done retroactively using the commit hash. More information can be found at [Git - Tagging](https://git-scm.com/book/en/v2/Git-Basics-Tagging).

* To tag the current version of the main branch
  + *git tag –a* <tag name> *-m “*< message about what the tag refers to>” [create tag]
  + *git push origin* <tag name> [upload tag to Github/Gitea]
* Example of above:
  + git tag –a v.202503 -m “Version used to produce March 2025 extract and exclusions”
  + git push origin v.202503
* To tag an earlier version of the main branch (i.e. if you’ve forgotten)
  + **NB: always do this based on the commit hash that relates to the merging of a feature branch (i.e. March run) with the main**
  + *git tag –a* <tag name> <commit hash> [create tag]
  + *git push origin* <tag name> [upload tag to GH/Gitea]

**Undo a commit**

If you have made a commit and you want to go back, you can undo your most recent commit by using git reset. This should only be done if you have not pushed your commits to remote.

* To undo the most recent commit, you can use ‘HEAD’ to reference the current commit (latest) on the current branch
  + *git reset HEAD~*
* To get a list of your most recent commits and git updates
  + *git reflog*
    - This will highlight the head by ‘HEAD ->’
* If there are more than one commit on your local branch, which are NOT pushed to remote you can select the specific commit by referring to the numbered HEAD, i.e.
  + *git reset HEAD~3*