**GIT/GITHUB**

**GIT**

Git is a programme that lets you keep track of changes and files. This includes managing source code. Download from the *gitbash* website.

Basic commands

|  |  |
| --- | --- |
| **Commands** | **Description** |
| Git –version | See current git version |
| Git config --global user.name “name” | Set a username |
| Git config --global user.email “email” | Set email address |
| Git config --list | To check if above cmds saved |
| Clear | Clear screen |

Creating a repository

Create a folder on local computer with or without files in.

*How to navigate round git:*

|  |  |
| --- | --- |
| **Commands** | **Description** |
| Pwd | To see current directory |
| Cd | To change directory (follow this with folder path) |
| Cd .. | Move back one folder |
| Ls | View all files/folders in current directory |
| Git init | Initialise – this creates repository (make sure you are in correct directory prior to this cmd) |

Initialise folder using *git init* cmd.

Making changes – commit

|  |  |
| --- | --- |
| **Commands** | **Description** |
| Git add . | Add all changed files to staging area |
| Git add <filename> | Add specific changed file to staging area |
| Git commit -m “message” | Commit changes to the repo folder |
| Git log | View the commit history |
| Git log --author = “name” | To filter log by author name |
| Git status | Tells you if any files modified or not yet committed |

Save files locally, then on *gitbash* you can *add* to staging area before *committing* changes.

Viewing changes made

|  |  |
| --- | --- |
| **Commands** | **Description** |
| Git diff | Shows differences between repo copy and local copy |
| Git diff --staged | Shows differences between repo copy and staged copy |

Deleting, renaming and moving files

|  |  |
| --- | --- |
| **Commands** | **Description** |
| Git rm <filename> | Removes file - \*this needs to be committed to be confirmed\* |
| Git add <new filename> | Rename file as normal locally and then add the new name using this cmd – then commit. Ensure to remove old file name using above method |
| Git mv <filename> <new filename> | Another way to rename file using gitbash only. (no need to change name locally) |
| Git mv <filename> folder/ | Move file to a specific folder |
| Git checkout -- <filename> | If you are making changes and have saved locally, but want to revert back, use this cmd. |
| Git reset HEAD <filename> | Unstage file |

Getting old versions from the repository

|  |  |
| --- | --- |
| **Commands** | **Description** |
| 1. Git log | To view all previous commits |
| 1. Git checkout (4+ characters of commit no.) --<filename> | This refers back to selected commit |
| 1. Git commit -m “message” | Commit changes |

**GITHUB**

GitHub allows you to make public projects. It is an open source where you can upload or download files. People can collaborate and work on projects together.

How to use github

To create a github repository:

* On github website click ‘new repository’ and fill in screen before ‘create repo’
* On gitbash – git remote add (nickname) <github repo link>
* Git push -u (username) master

Git push – pushes the repo file to server.

Github desktop

Download github desktop for GUI instead of using gitbash, if preferable.

To commit changes on github desktop – adjust files locally as normal, then on GUI *commit to master > sync*

This can now be seen on github website. Click #commits on github website to view all commits of repository.

Branches

Branches basically are a copy of a main/master repository – this is used when editing files.

To create one:

1. on github website click on the drop down ‘*branches’* and create a name.
2. edit, upload or create new files
3. **To confirm changes:** click branches tab and select ‘new pull request’ on specific branch and enter title & description to describe changes made
4. click ‘merge pull request’ to update master branch with this branch
5. You can now delete this branch if not needed

Github extras

The watch, star and fork buttons:

* Watch = watch a project (follow a project)
* Star = favourite a project
* Fork = make a new branch/copy to edit yourself

The issues tab is similar to a to-do-list. You can assign people for specific tasks and create tasks for them to complete. These can be labelled with ‘labels’ such as *bug, enhancement…*

You can create github wiki for your project, which you can use to add info about your project for viewers viewing your project. To do this: click *wiki* and then *create first page*.

You can also create github organisations via your profile for your company – similar to a facebook group. You can then create teams in these organisations which can give a group of people certain permissions/restrictions.