**GIT**

* git commands won't work outside a Git repo.
* Whenever you exit a repo directory, be sure to change back into the directory with cd before issuing any Git commands.
* With Git, every developer has their own repository on their machine, called their local repo.
* Repos belonging to other developers on the project are referred to as remote repos.
* Each repo is a full copy of the original repo, including all the version history.
* Developers can make commits to their own repo, independently of the others.
* Before beginning work on a new feature, you should use the network to pull commits other developers have made from a remote repo into your local repo, so that you have the latest version of the code.
* Then you can make your changes, and commit them to your local repo.
* Other developers will be able to pull the commits *you've* made over the network to their clones, so they can access them.

**File statuses**

There are three states every file goes through in a Git repository.

* Modified – changes made to a file in the working directory
* Staged – files added to the index or staging area of your next commit
* Committed – a version of your file stored in a repository

**Git Command examples** – all in the format git example

N.B. HEAD can be used on any commands that take a SHA value from the log. HEAD means use the most recent commit message.

* Status ~~ list the files you’ve changed and those still need to add or commit
* Init ~~ create a new local repository
* Add examplefile.txt ~~ adds file to the staging area (index)
* Commit –m “Needs an info quote” ~~ adds files from the staging area to the repository
* Commit ~~ this opens a Commit text editor to manually write your commit
* Config –global user.email “example.email.com” ~~ this sets your email
* Config –global user.name “John Smith” ~~ this sets your user name
* Log ~~ shows the commit logs and their Simple Hashing Algorithm (SHA), press Q to exit log
* log –p ~~ as above but more detailed
* diff ~~ compares current contents of files to those in the staging area and show what’s changed
  + New lines will show with a + sign in front of them.
  + Removed lines will show with a - sign in front of them.
  + And changed lines will show with a - sign in front of the old version, and a + sign in front of the new version.
* Diff –staged ~~ compares your staged changes with a previous commit
* Rm <file> ~~ flags the file for removal in the staged area and will be removed on the next commit e.g. git commit –m “Removed examplefile.html”
* Mv <file> <file> ~~ move is similar to rename e.g. git mv example1.txt example1.html on next commit
* Reset HEAD <file> ~~ removes modified filed from staged if they have not yet been committed e.g. git HEAD example1.txt
* Checkout **--** <file> ~~ discards changes to a modified file, i.e. makes it the same as the last committed version e.g. git checkout example1.txt
  + The above command also works if a file has been accidentally deleted from the working directory. Running it will recover the file and all its contents.
* Revert ~~ undoes previous commits. You can use HEAD for most recent or first 5 or so SHA characters from the relevant commit in the log. E.g. git revert 1d8el or git revert HEAD.
  + This will open up the commit editor to make any relevant changes
* Clone ~~ make an exact copy of a directory (repo) e.g. to copy a local directory you would type git clone repo1 repo2. N.B. to copy a remote repo you would use the full URL and clone it in to a local directory. By default it would be called the same but you can put a new name after the URL if required.
* Pull ~~ pulls commits from a remote repo
  + Within a Git repository, you can add links to other repos. These linked repos are referred to as remote repos.
  + We can get a list of remote repos with the git remote command.