Version control with Git (/git-course/): Glossary

Key Points

Introduction (/git-course/01-introduction/index.html)	• Git is a version control tool; one of many. • GitHub is a repository hosting service; one of many. • Use version control to store versions neatly, restore previous versions, understand what happened (and why), and always know which is the current version.
Tracking changes with a local repository (/git-course/02-local/index.html)	• `git init` initializes a new repository • `git status` shows the status of a repository • Files can be stored in a project's `working directory` (which users see), the `staging area` (where the next commit is being built up) and the `local repository` (where commits are permanently recorded) • `git add` puts files in the staging area • `git commit` saves the staged content as a new commit in the local repository • Always write a log message when committing changes
Looking at history and differences (/git-course/03-history/index.html)	• `git log` shows the commit history • `git diff` displays differences between commits • `git switch -d` recovers previous states of the repo • `HEAD` points to the commit you have checked out • `master` points to the tip of the `master` branch
Commit advice (/git-course/05-commit-advice/index.html)	• Commit messages explain why changes were made, so make them clear and concise • Follow conventions to give a history that is both useful, and easy to read • Only commit files which can't be automatically recreated

Branching (/git-course/06-branching/index.html)	`git switch -c <branch_name>` creates a new branch - Use feature branches for new ideas and fixes, before merging into `master` `git merge <branch_name>` merges into current branch - Merging does not delete any branches
Undoing changes (/git-course/08-undoing/index.html)	`git restore <file>` discards unstaged changes `git commit --amend` allows you to edit the last commit `git revert` undoes a commit, preserving history `git reset --hard` undoes a commit by deleting history
Working from multiple locations with a remote repository (/git-course/09-remote/index.html)	- Git is the version control system: GitHub is a remote repositories provider. `git clone` to make a local copy of a remote repository `git push` to send local changes to remote repository
Collaborating with a remote repository (/git-course/10-remote-collaboration/index.html)	`git pull` merges remote changes into local branch of repository
Rebasing (/git-course/12-rebasing/index.html)	`rebase` applies your changes on top of a new base (parent) commit - rebasing rewrites history
Pull Requests (/git-course/13-pull-requests/index.html)	- A `fork` is a `git clone` into your (GitHub) account - A `pull request` asks the owner of a repository to incorporate your changes - Use issues and GitHub projects to plan your work - You can discuss code on both issues and pull requests
Git hints and tips (/git-course/14-hints-and-tips/index.html)	- List files to ignore by committing a `.gitignore` file - Selectively stage changes to files using `git add --patch` - Configure a visual diff tool and use `git difftool` to view changes
Conclusions and further information (/git-course/15-conclusion/index.html)	Use version control whenever possible

Glossary

FIXME

Edit on GitHub (https://github.com/gcapes/git-course/edit/gh-pages/reference.md) / Contributing (https://github.com/gcapes/git-course/blob/gh-pages/CONTRIBUTING.md) / Source (https://github.com/gcapes/git-course/) / Cite (https://github.com/gcapes/git-course/blob/gh-pages/CITATION) / Contact (mailto:its-research-training@manchester.ac.uk)

Using The Carpentries style (https://github.com/carpentries/styles/) version 9.5.3 (https://github.com/carpentries/styles/releases/tag/v9.5.3).