GIT 🡪 Global Information Tracker

* Version control software/source control
  + Manage changes made to files overtime
  + Document history of a project, with the ability to jump back and forward through versions (branches)
  + Check points (commits)
  + Branches 🡪 create alternate versions of your code. Can work on with or without changing the original
  + Synchronize (merge)

Setting up a project:

First step – setting up configuration variables.

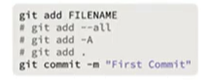
* GIT is designed to be used by multiple people – need to tell it who you are so it gives you credit for changes you make
* $ git config --global user.email --replace-all "michael.sturman@bath.edu"

Next prepare the folder you’d like to contain the project (see learngithub in the attached)

Open in visual studio and launch a new terminal – select git bash.

Run the git init command to initialize an empty git repository

In order to create a save we can go back to we have to add the files to the staging environment. Use the add command.



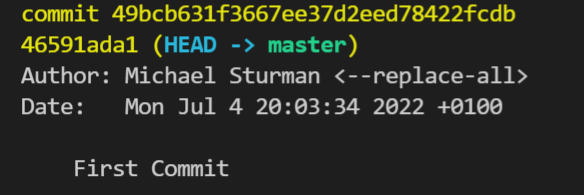
Staging is a temporary area we can store files we want to commit later on.

-- all adds all files in the project (-A is the shortcut). Most longer commands have shortcuts that 🡪 first letter of the command (period is a short cut for the current directory, also works).

Git commit -m “First Commit”

Tells git that this is one of the checkpoints we want to track for our project. We can come back to it later.

Understanding Environments



Commit hash at top 🡪 unique entry for the commit. Next to it tells us the HEAD is currently in the main branch. Branches are used to organize the project. Each branch is like an “alternate reality” of the project. The HEAD always points to the current branch we are working on.

Git Environments:

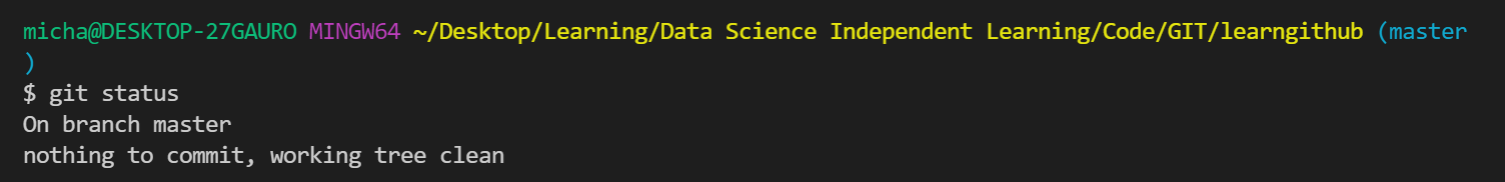
3 places to put files:

* Working – where the files look like they did after the last commit
* Staging – where you move files before a commit (temporary location). Queue up changes before you’re ready to commit
* Commit – new log entry is created with a new hash

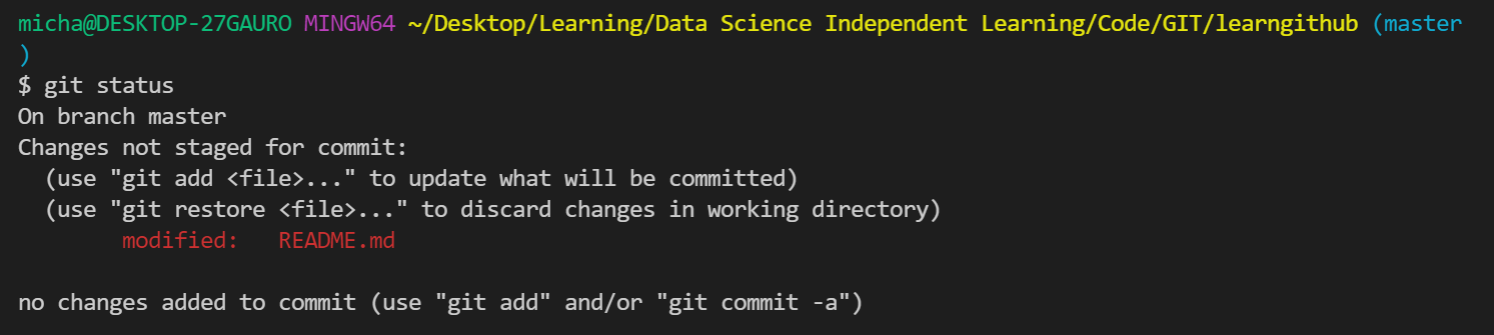
File States:

* Tracked: files that existed in prev snapshot
  + Unmodified: files haven’t changed since last commit
  + Modified: have changes
  + Staged
* Untracked: anything else (eg new file added since the last commit)

Viewing Status (git status) shows what a files current status is



Before an un-staged/committed change

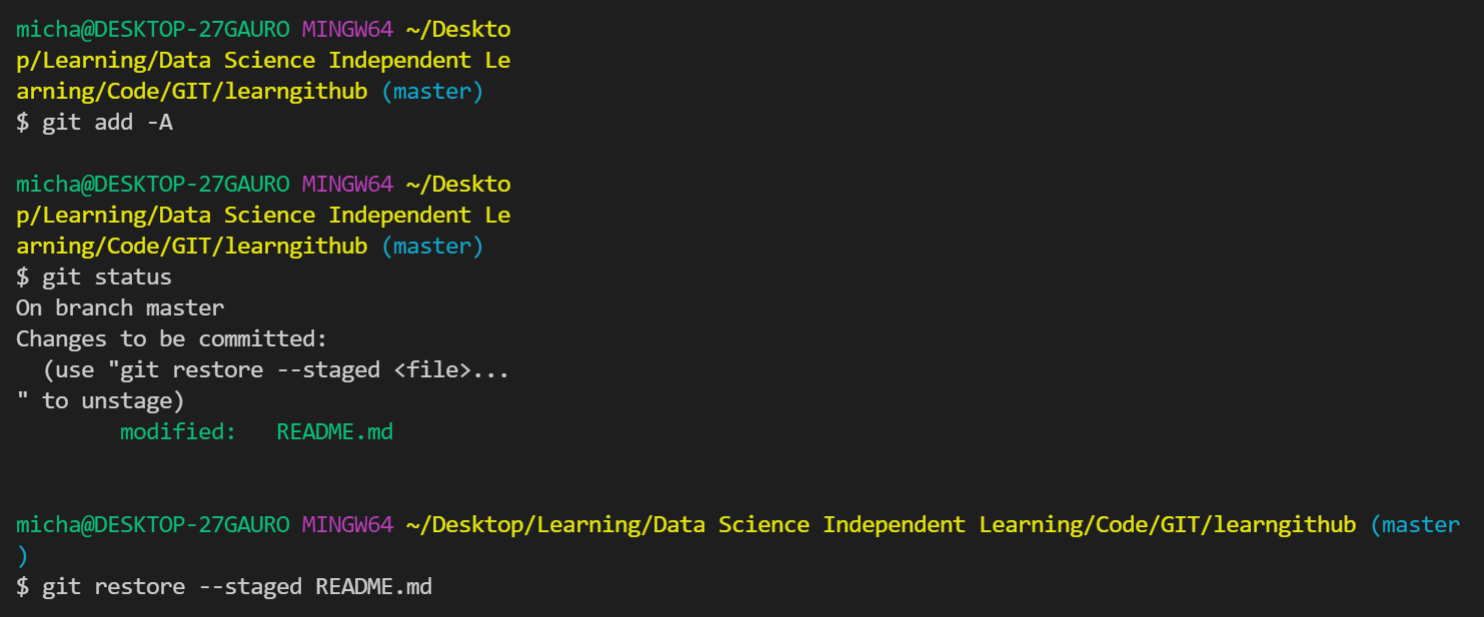


After an un-staged/committed change

Now we have options – use the add command to add the changes to staging (as above) or use restore to undo the changes.



Similarly to above, can restore by file name or by directory.



Above for restoring a staged file