

Introduction to git

- git is a distributed revision control system
- useful for managing text based changes across multiple files and directories
 - distributed
 - allows files to be removed, moved
 - user can commit changes and retrieve previous changes
- very useful for programming or managing documents in `groff` or `latex` format

Introduction to git

- not so useful for `openoffice` documents
- but it could still be used
 - just not as efficient as using text

Server side: setting up your own git server

- use `mcgreg.comp.glam.ac.uk`

```
$ cd
$ mkdir myrepo.git
$ cd myrepo.git
$ git --bare init
```

- the above only needs to be done once

Client side

- firstly we need to define where the git repo lives
- ```
$ git remote add origin username@mcgreg.comp.glam.ac.uk:/
$ git push origin master
```
- defines `origin` to the `mcgreg` url
- sets the master branch (like `cvs` head) to the origin
  - for personal git repos you will rarely need to bother with branches
  - thus the `master` branch can be ignored here

## Client side

- ```
$ mkdir -p Sandpit/firstcopy
$ cd Sandpit/firstcopy
$ git init
$ git add
$ touch README
$ git commit -m "My first file committed"
$ git push
```

Client side copy

- to take another copy of the files from the repository you can

- ```
$ mkdir -p Sandpit/secondcopy
$ cd Sandpit/secondcopy
$ git clone username@mcgreg.comp.glam.ac.uk:/home/username
```

- now you could make some modifications and then

- ```
$ cd Sandpit/secondcopy
$ emacs README
$ git commit -m "My second commit"
$ git push
```

- which uploads changes back to the master on mcgreg

What changes have I made?

- can use

- ```
$ git diff
```

- to see the differences between your local copy and the repository

## What changes have I made?

- can use:

- ```
$ git log
```

- to see the entire history of all changes, or:

- ```
$ git log version.c
```

- to see the history of all changes to the file version.c

## Tagging in git

- you can tag a release or stage of development by:

- ```
$ git tag version1.0 -m "version 1.0"
```

- to see all available tags type:

- ```
$ git tag
```

## To go back in time to a previous version

- ```
$ mkdir thirdcopy
$ cd thirdcopy
$ git clone username@mcgreg.comp.glam.ac.uk:/home/username
$ git checkout version1.0
```

To go back in time to a previous version

- or you could go back to another commit time (as seen by the `git log`)

- ```
$ mkdir fourthcopy
$ cd fourthcopy
$ git clone username@mcgreg.comp.glam.ac.uk:/home/username
$ cd myrepo.git
$ git log
etc etc
ec1c3b3fa9844e3304fe24eb54e0529e356883f7
etc etc
$ git checkout ec1c3b3fa9844e3304fe24eb54e0529e356883f7
Note: checking out 'ec1c3b3fa9844e3304fe24eb54e0529e356883f7'.
You are in 'detached HEAD' state. You can look around, make
changes and commit them, and you can discard any commits :
state without impacting any branches by performing another:
```