Version control

- what is version control?
- setting up Git
- simple command-line usage
- Git basics

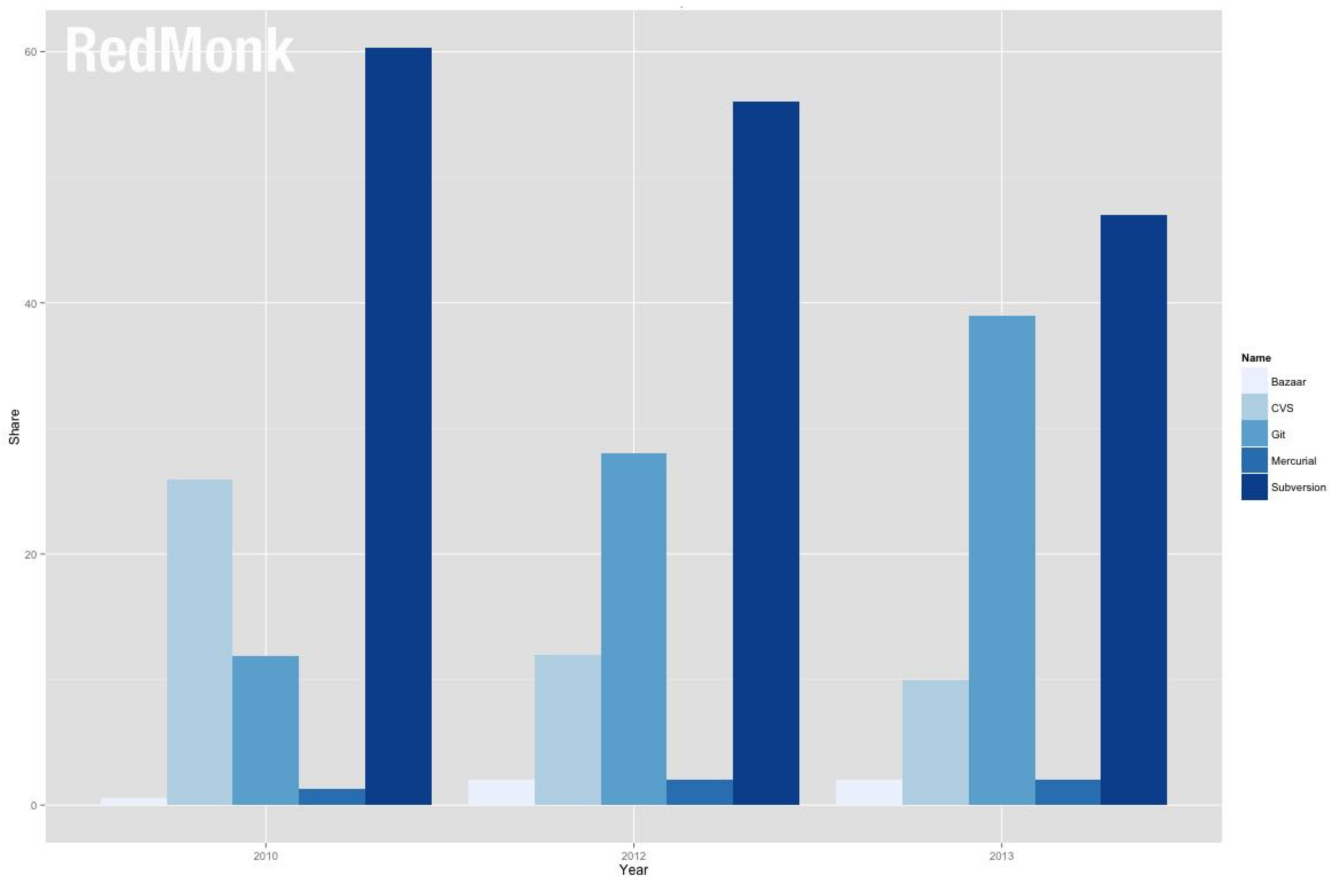
Version control - intro

- ensure we keep track of changes, updates, contributions, suggested mods...
- could try old, and error-prone, tracking of directories etc
- *version control* tool such as **Git**
- coding style known as *exploratory coding*
  - *researching, learning, checking what does and does not work correctly...*
  - *often used methodology for coders, and small groups as well*
- can lead to many changes and updates in code

Version control - what is version control?

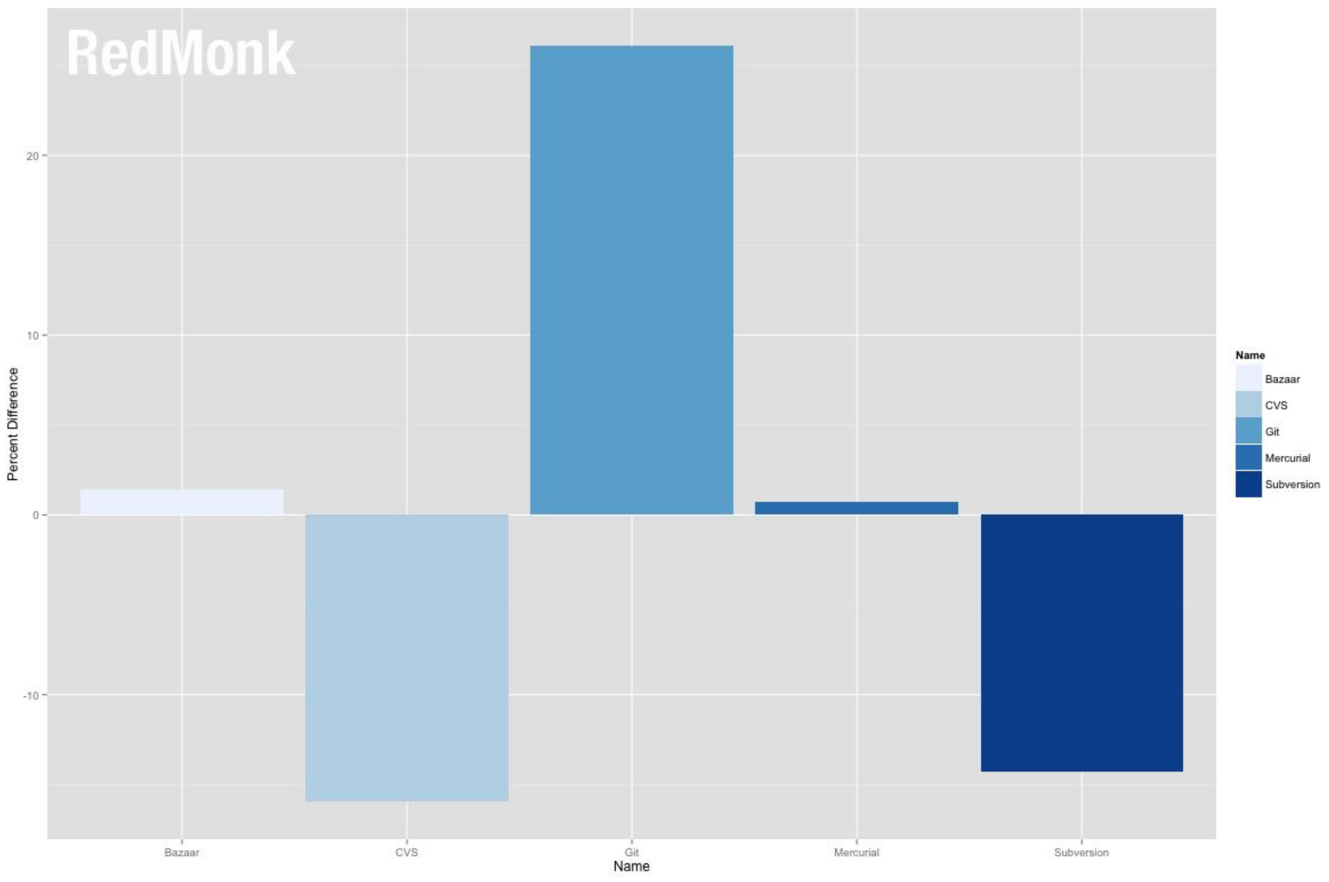
- very basic form of version control used on a regular basis
  - *copying, replicating folders, documents etc*
- compare updates between old and new copies & revert back to older version
  - *very basic form of version control*
- software development version control
  - *maintain defined, labelled points in our code*
  - *easily refer back to them or revert to an earlier state if needed*
  - *important tool for collaborative work with other developers*
- number of different, and popular, version control tools over the last few years
  - *Subversion, Mercurial, Git...*
- by 2010 Subversion held approximately 33.4% of the market for version control
  - *Git is believed to have only held approximately 2.7%, and Mercurial a paltry 0.7%*
- by 2013, Git usage was almost as high as Subversion, and it continues to grow in popularity
- Git's popularity largely due to preference for distributed version control
  - *Atlassian's switch from Subversion to Git in 2012 also helped*

Image - Version control usage (2010-2013)



Source - RedMonk

Image - Version control change in usage (2010-2013)



Source - RedMonk

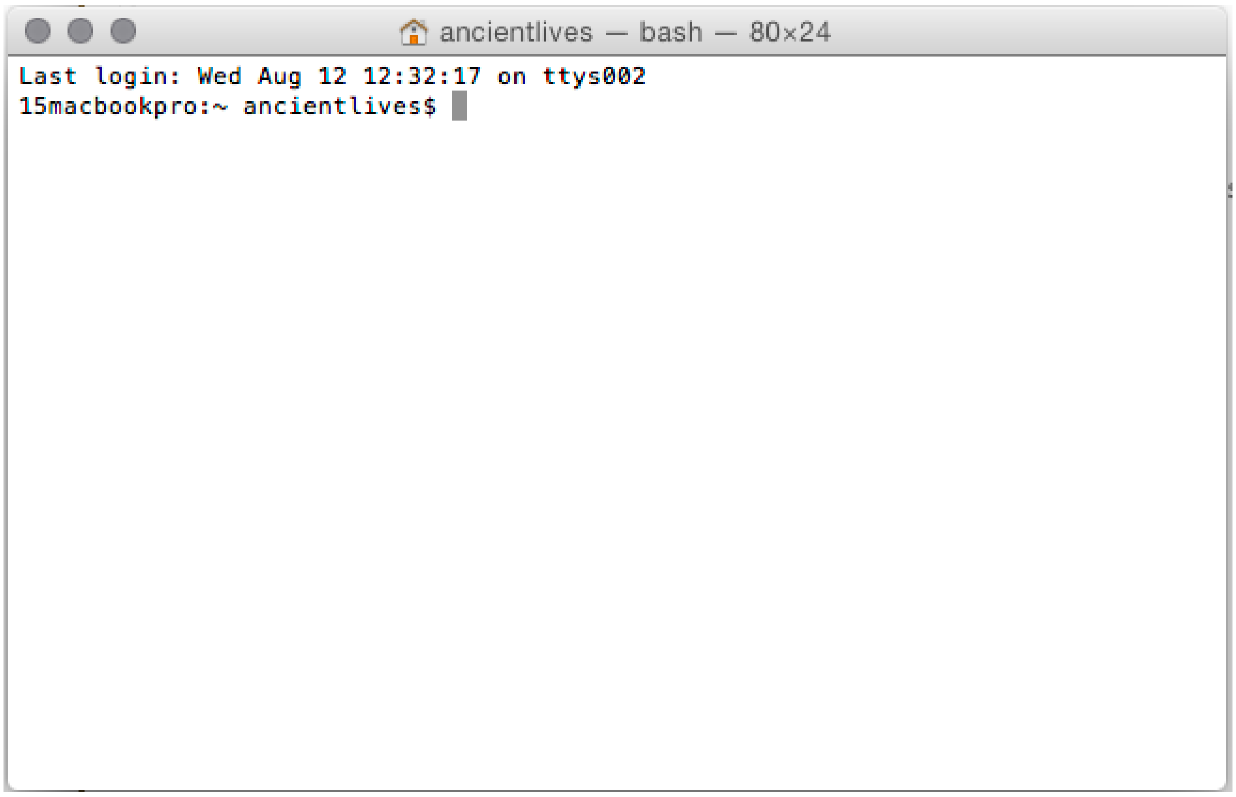
Version control - setting up Git

- simple installers available for Git
- choose platform's installer from
  - *git*
  - *follow simple instructions to install*
- full install instructions for various Linux distributions
  - *git - Linux downloads*
- for Debian/Ubuntu based APT distributions
  - `apt-get install git`

Version control - Git GUIs

- many GUIs available for working with Git
  - *Git GUIs*
- including specific GUIs for GitHub
  - *GitHub desktop clients*
- still beneficial and quicker to work with command-line
  - *quick and easy to navigate files, directories...*
  - *work with Git and version control in general*

Image - OS X Terminal application



```
Last login: Wed Aug 12 12:32:17 on ttys002
15macbookpro:~ ancientlives$
```

Command-line - Navigating directory and files

A few examples

- check the current directory (pwd = *print working directory*.)

```
pwd
```

- check the contents of the current directory (lists working directory)

```
ls
```

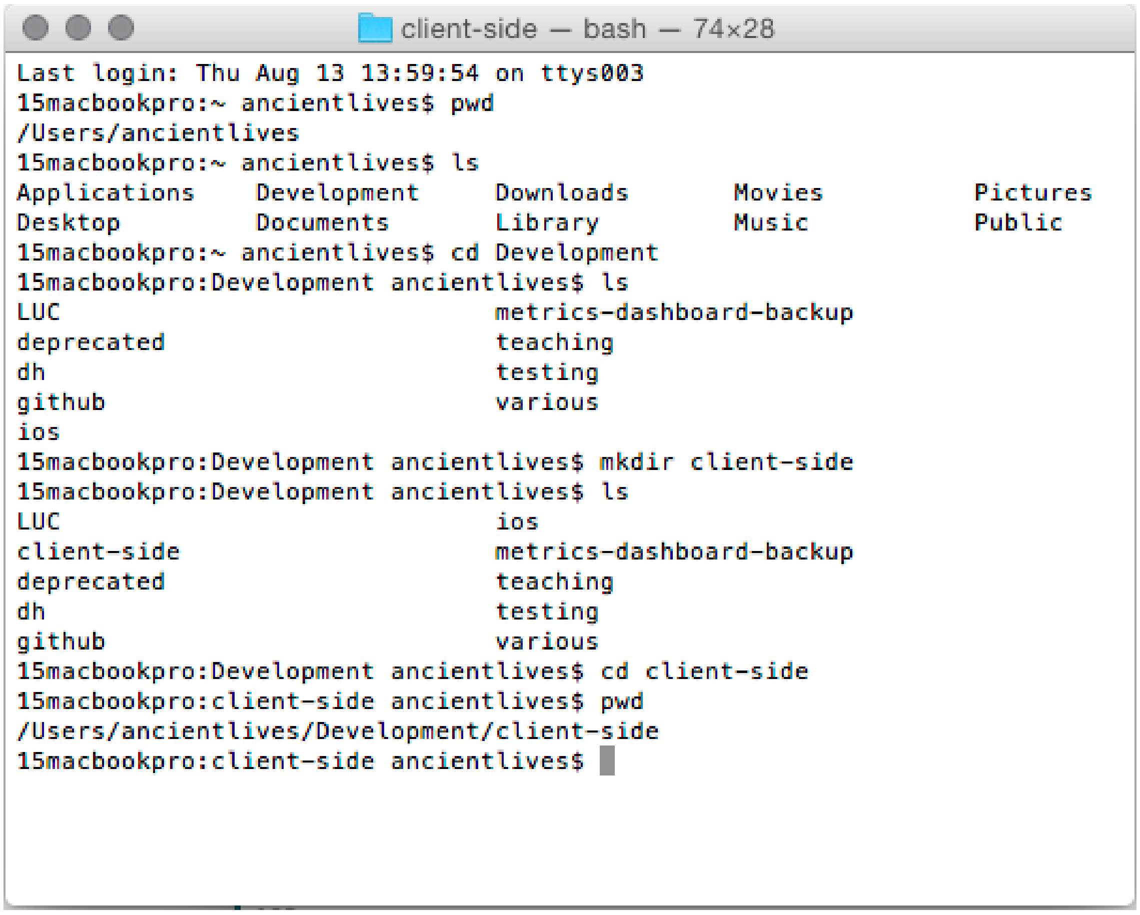
- this command allows us to change directory

```
cd
```

- in the working directory, we can create a new directory

```
mkdir
```

Image - Command-line examples



client-side — bash — 74×28

```
Last login: Thu Aug 13 13:59:54 on ttys003
15macbookpro:~ ancientlives$ pwd
/Users/ancientlives
15macbookpro:~ ancientlives$ ls
Applications    Development     Downloads       Movies          Pictures
Desktop         Documents       Library         Music           Public
15macbookpro:~ ancientlives$ cd Development
15macbookpro:Development ancientlives$ ls
LUC                             metrics-dashboard-backup
deprecated                      teaching
dh                              testing
github                          various
ios
15macbookpro:Development ancientlives$ mkdir client-side
15macbookpro:Development ancientlives$ ls
LUC                             ios
client-side                     metrics-dashboard-backup
deprecated                      teaching
dh                              testing
github                          various
15macbookpro:Development ancientlives$ cd client-side
15macbookpro:client-side ancientlives$ pwd
/Users/ancientlives/Development/client-side
15macbookpro:client-side ancientlives$ 
```

Configure user/developer details

- set details for *username* and *user email*
  - *global flag can set these details for all work within our local instance of Git*

```
git config --global user.name "424dev"
```

- set preferred email address

```
git config --global user.email "424dev@gmail.com"
```

- override for a specific repository in Git by omitting `--global` flag

```
git config user.name "424dev-single"
```

and the same principle applies for email.

- check correct username for current repository

```
git config user.name
```

Tracking projects

- start tracking project with Git
  - *create new working directory (eg: at `root` of our home directory)*

```
cd ~/
```

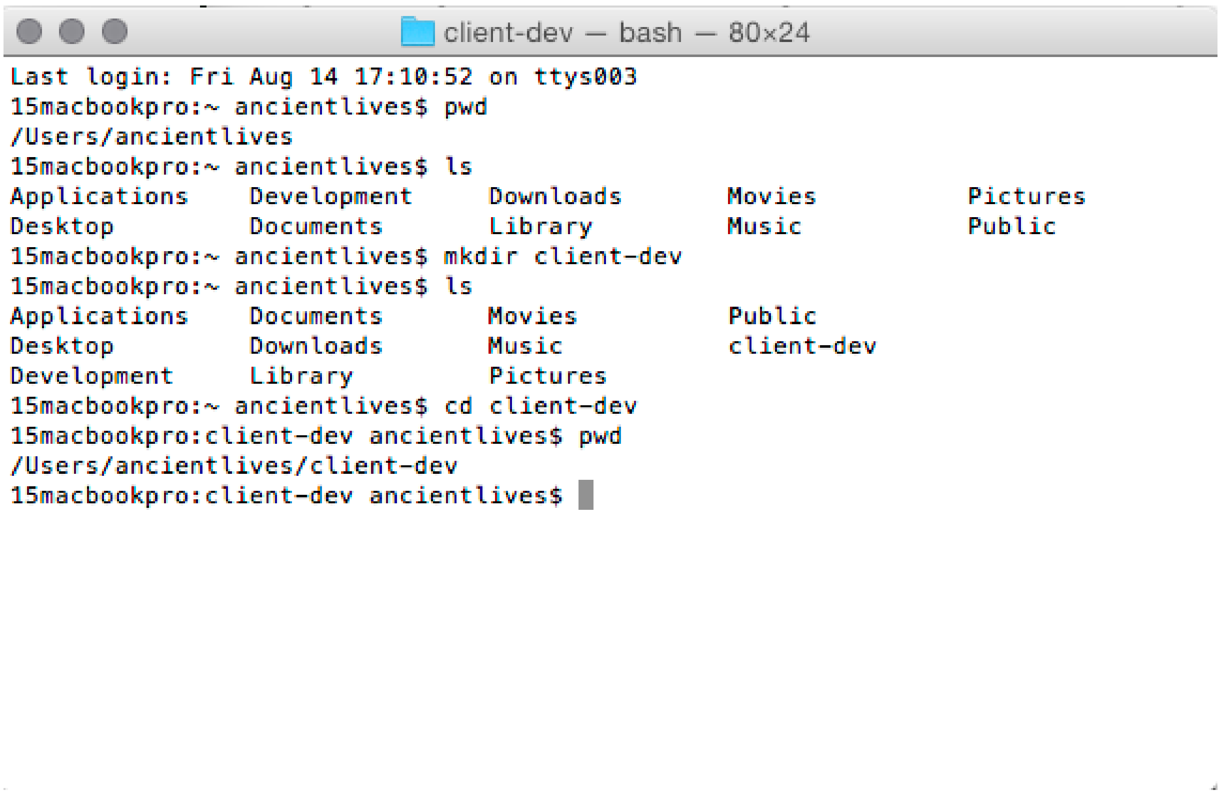
- ensures we have changed to our home directory. Then check working directory,

```
pwd
```

and then make a new directory for our client-side development.

```
mkdir client-dev
```

Image - creating a *client-dev* directory



```
Last login: Fri Aug 14 17:10:52 on ttys003
15macbookpro:~ ancientlives$ pwd
/Users/ancientlives
15macbookpro:~ ancientlives$ ls
Applications    Development     Downloads       Movies          Pictures
Desktop         Documents       Library         Music           Public
15macbookpro:~ ancientlives$ mkdir client-dev
15macbookpro:~ ancientlives$ ls
Applications    Documents       Movies          Public
Desktop         Downloads       Music           client-dev
Development     Library         Pictures
15macbookpro:~ ancientlives$ cd client-dev
15macbookpro:client-dev ancientlives$ pwd
/Users/ancientlives/client-dev
15macbookpro:client-dev ancientlives$ 
```

Git basics - Part 3

Add version control using *Git* to working directory

- initialise our new repository in the working directory

```
git init
```

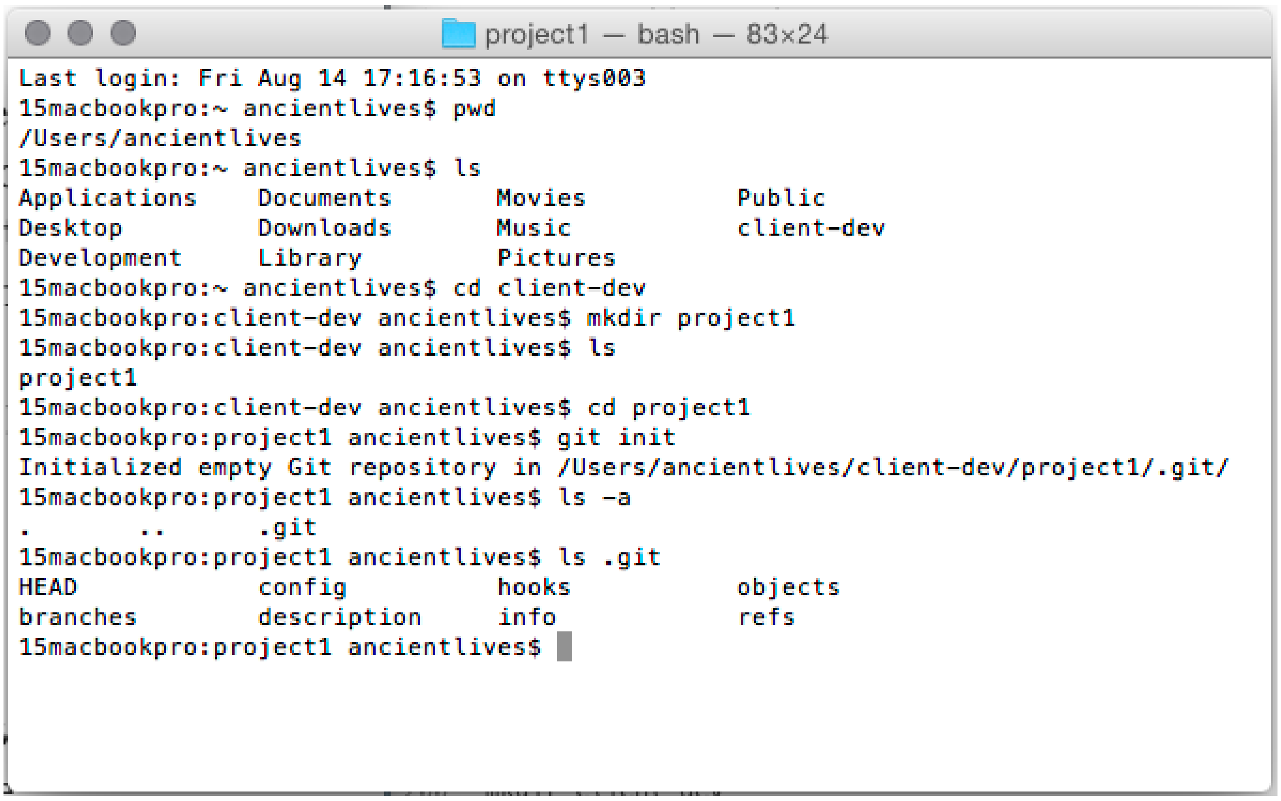
- check hidden `.git` directory has been created

```
ls -a
```

- and show contents of the `.git` directory

```
ls .git
```

Image - Initialise new Git repository



project1 — bash — 83×24

```
Last login: Fri Aug 14 17:16:53 on ttys003
15macbookpro:~ ancientlives$ pwd
/Users/ancientlives
15macbookpro:~ ancientlives$ ls
Applications    Documents       Movies          Public
Desktop         Downloads       Music           client-dev
Development     Library         Pictures
15macbookpro:~ ancientlives$ cd client-dev
15macbookpro:client-dev ancientlives$ mkdir project1
15macbookpro:client-dev ancientlives$ ls
project1
15macbookpro:client-dev ancientlives$ cd project1
15macbookpro:project1 ancientlives$ git init
Initialized empty Git repository in /Users/ancientlives/client-dev/project1/.git/
15macbookpro:project1 ancientlives$ ls -a
.       ..      .git
15macbookpro:project1 ancientlives$ ls .git
HEAD            config          hooks           objects
branches        description     info            refs
15macbookpro:project1 ancientlives$ 
```

Git basics - Part 4

Start using our new repository

- create an initial `index.html` file in project's working directory
  - *create using preferred text editor or command-line, eg:*

```
touch index.html
```

- save new file, and check *Git* is correctly tracking our project

```
git status
```

- outputs current status of working branch, defaults to `master`
  - *outputs we have `untracked files`*
  - *files will include new `index.html`*
- add any new untracked file/s

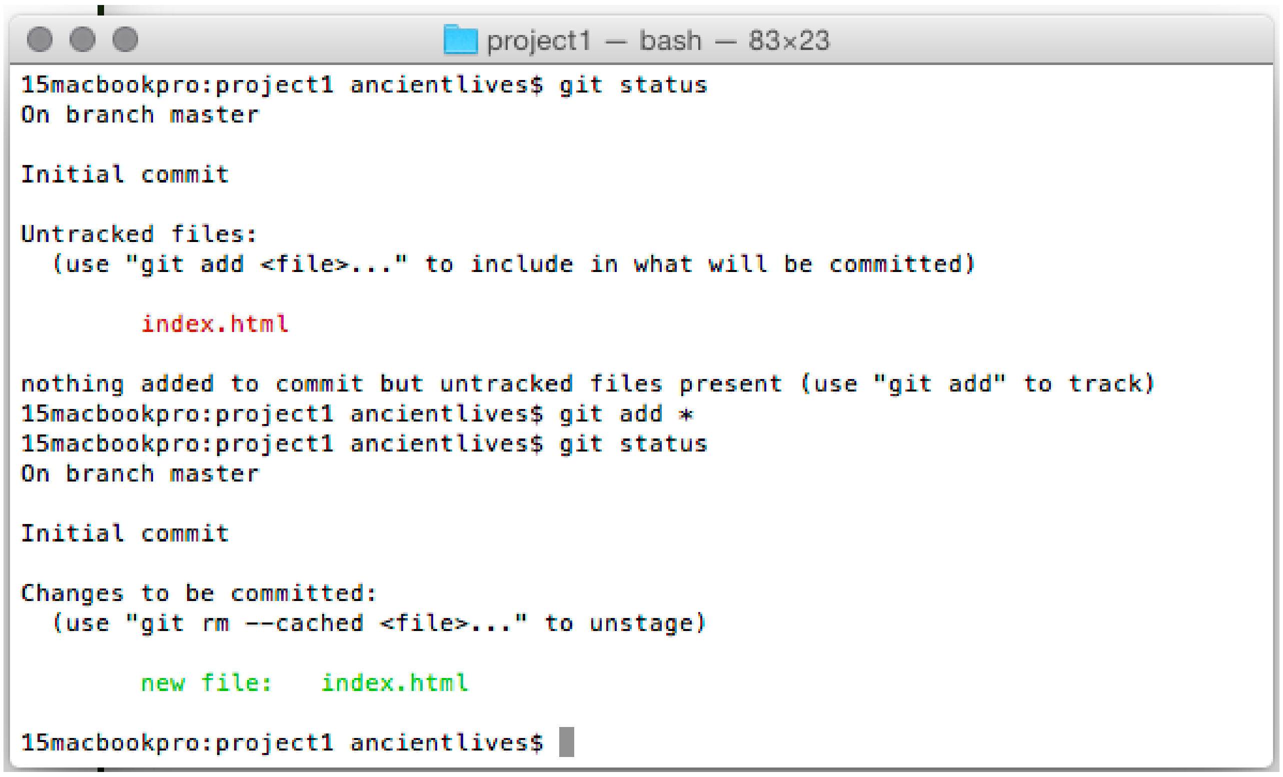
```
git add index.html
```

or

```
git add *
```

for all untracked files.

Image - Git status and add



```
15macbookpro:project1 ancientlives$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        index.html

nothing added to commit but untracked files present (use "git add" to track)
15macbookpro:project1 ancientlives$ git add *
15macbookpro:project1 ancientlives$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   index.html

15macbookpro:project1 ancientlives$
```

Git basics - Part 5

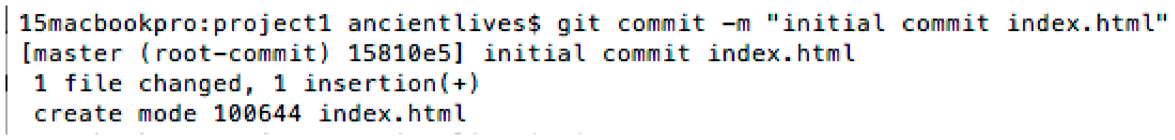
After adding our new `index.html` file to the repository, we can commit these changes to the initial state of the repository.

We use the following command

```
git commit -m "initial commit index.html"
```

- `–m` flag permits useful message for commit
  - *message added within quotation marks*
  - *should be useful and present tense*

Image - First commit and message



```
15macbookpro:project1 ancientlives$ git commit -m "initial commit index.html"
[master (root-commit) 15810e5] initial commit index.html
 1 file changed, 1 insertion(+)
 create mode 100644 index.html
```

Git basics - Part 6

- initial commit has saved a defined point in our work
    - *one we can revert to if needed*
- check `git status` and there should be nothing else to commit
    - *working directory should be clean*
- *Git* has set our files ready for tracking
- repeat this process as we make further changes and updates
    - *freeze defined points within our project*
- check recent commits, and view a record

```
git log
```

Git basics - Part 7

Git revisions

- we've seen *Git*'s simple linear commits
  - *presumes file has one parent*
  - *child commits detail this linear revision of content*
- a *Git* commit can store multiple parents and children
- eg: commit history might include
  - *revisions to a single file*
  - *addition or deletion of new files*
  - *merging of files to different branches*
  - *further additions...*

Git basics - useful commands

Git command	Expected Outcome
git config user.name "..."	sets username for current repo
git config –global user.name "..."	sets username for all repos (unless overridden per repo)
git config user.email "..."	sets user's email address for current repo
git config –global user.email "..."	sets user's email address for all repos (unless overridden per repo)
git init	initialise a Git repository in the current working directory
git status	output current status of repo in current working directory
git add "..."	define specific file in current repo for next commit
*git add **	define all files in current repo for next commit
git commit -m "..."	commit defined files (set using `git add`) with message
git log	output commit history for current repo