

# Version control

(using git)

---

**Danny Awty-Carroll**

May 15, 2018

**IBERS**

Athrofa y Gwyddorau Biolegol, Amgylcheddol a Gwledig  
Institute of Biological, Environmental and Rural Sciences

# Sections

This will focus on using git in windows with a UI

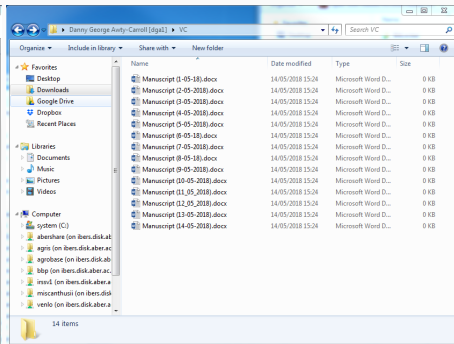
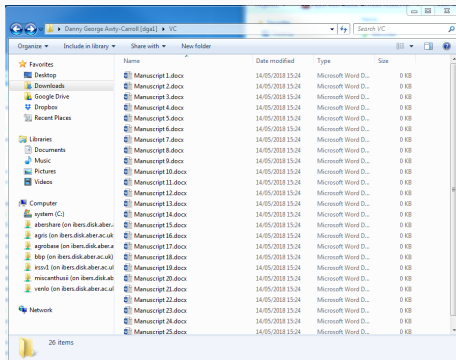
1. Why version control?
2. Version control system
3. Platform
4. UI
5. Using git
6. Questions & Demo

# Why version control?

---

# What is version control

This is just the management of versions of a document.  
One document through time.



All of us use some version control

# Where things get complicated

Numbering or dating documents works OK but can fall down when

- There are multiple documents that need to work together (i.e. a script and data)
- There are multiple people working on the documents (are they on the latest version and merging changes)
- There are updates to the project that may break things
- Line changes need to be reviewed

Some of this is solved in programs like word with track changes (so you see who altered what when)

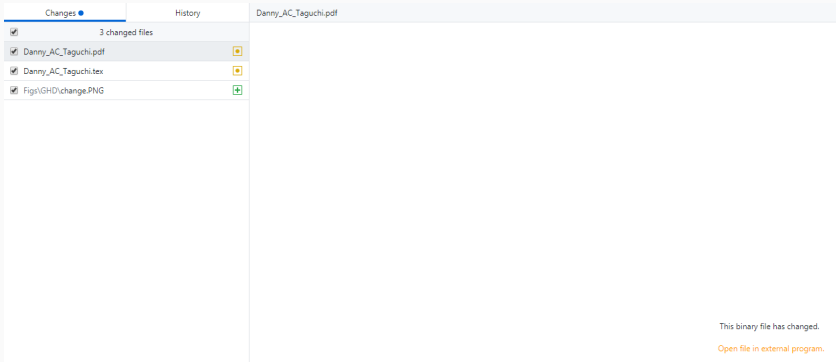
# What version control systems can do

- Give line by line changes linked to a user (which are kept forever)
- Makes a stretcher to the version control so there can be side branches (this is where the project can take a detour and be merged back in later)
- Minimise problems of multiple users working on the same document at the same time

| Changes ●            | History | Danny_AC_Taguchi.tex  |
|----------------------|---------|---|
| 2 changed files      |         | @@ -72,14 +72,14 @@ Some of this is solved in programs like word with track changes (so you see who                 |
| Danny_AC_Taguchi.pdf |         | 72 72<br>73 73 \begin{frame}[fragile]{what version control systems can do}  |
| Danny_AC_Taguchi.tex |         | 74 74 \begin{itemize}   |
|                      |         | 75 - \item There are multiple documents that need to work together (i.e. a script and data)                         |
|                      |         | 76 - \item There are multiple people working on the documents (are they on the latest version and merging changes)  |
|                      |         | 77 - \item There are updates to the project that may break things   |
|                      |         | 78 - \item Line changes need to be reviewed   |
|                      |         | 75 + \item Give line by line user by user line changes  |
|                      |         | 76 + \item Make a stretcher to versions so there can be side branches   |
|                      |         | 77 + \item Minimise problems of multiple users working on the same document at the same time                        |
|                      |         | 78 \end{itemize}  |
|                      |         | 79 79 \end{frame}   |
|                      |         | 80 80   |
|                      |         | 81 80   |
|                      |         | 82 - \begin{frame}[fragile]{what version control systems can do}  |
|                      |         | 81 +  |
|                      |         | 82 + \begin{frame}[fragile]{what version control systems can do}  |
|                      |         | 83 83 \begin{itemize}   |
|                      |         | 84 84 \item There are multiple documents that need to work together (i.e. a script and data)                        |
|                      |         | 85 85 \item There are multiple people working on the documents (are they on the latest version and merging changes) |

# What version control systems can't do

- Add much extra control to binary files (not plan text)
- Back up in real time



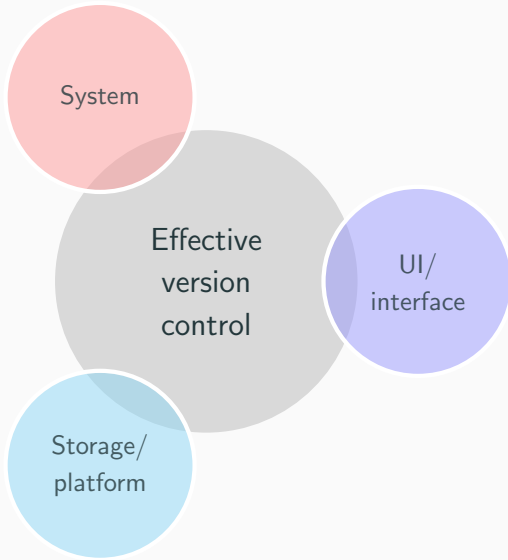
# Version control system

---



# What version control system?

The first thing about version control systems is which system (we will cover git)



# What version control system?

- There are two popular version control systems (git SVN)
- We will cover git as it is the most popular, the one I know and easiest to use
- Git was developed in 2005 by Linus Torvalds

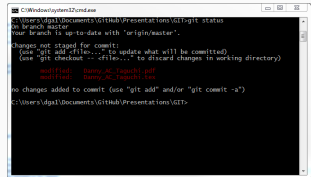


(Principal developer of Linux)

# What is git?



- Git is just system of version control

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window shows the output of the command "git status". The output indicates that the branch is up-to-date with 'origin/master', and there are no changes staged for commit. It lists two modified files: "dummy\_AC\_Tapochi.pdf" and "dummy\_AC\_Tapochi.tex".

```
C:\Windows\system32\cmd.exe
C:\Users\dgal\Documents\GitHub\Presentations\GIT>git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   dummy_AC_Tapochi.pdf
        modified:   dummy_AC_Tapochi.tex

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\dgal\Documents\GitHub\Presentations\GIT>
```

- By default it is used through a terminal
- Git has projects with multiple documents called repositories (or repos)
- It then has a tree structure to manage the versions of the repository

# Tree structure?



[master] 6c6faa5 My first commit - John Doe

[develop] 3e89ec8 Develop a feature - part 1 - John Doe

[develop] e188fa9 Develop a feature - part 2 - John Doe

[master] 665003d Fast bugfix - John Fixer

[myfeature] eaf618c New cool feature - John Feature

[master] 8f1e0e7 Merge branch 'develop' into 'master' - John Doe

[master] 6a3dacc Merge branch 'myfeature' into 'master' - John Doe

0.1

[master] abcdef0 Release of version 0.1 - John Releaser



The master branch is in grey with colour branches coming of and then being merged back

# Key commands?

In the terminal interface there are some useful commands:



- *init* - initialise a repo
- *status* - tells you if you have files out of sync with the current version
- *commit* - adds commits to the current version - a change with a comment and an ID
- *add* - adds new files and current commit
- *remove* - removes files and current commit
- *push* - pushes commits to an online repo (only if using a git platform)
- *pull* - pulls commits from an online platform (only if using a git platform)

THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.

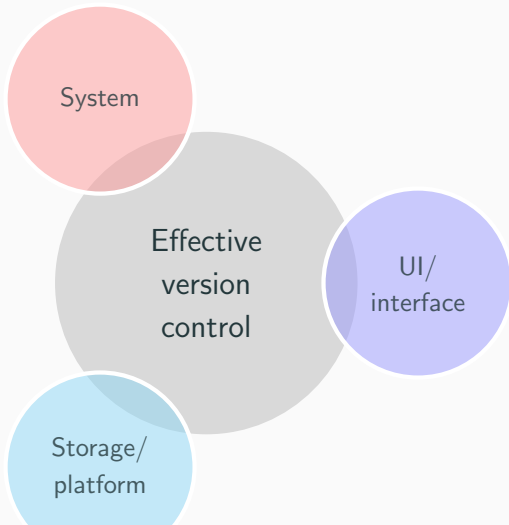


# Platform

---

# Which platform?

You can just use git with the terminal on your PC or you can store work on an online repository managing platform, the most popular of these are GitHub and Bitbucket





# Why not just store the repository on Dropbox?

You can and it would be backed up and sharable, but there are problems:

- You can run into Dropbox syncing and git version control clashes
- If you share the files with a collaborator they aren't identified separately to you
- You or a collaborator can completely mess up the git system (mostly by deleting the records in the '.git' folder)
- If you do you can't just re-download



# Which platform?

## Bitbucket

- Allows unlimited public or private repositories
- Charges per collaborators over 5 on those repositories
- 1GB storage for **large** files all repositories, 5GB if using a .ac email



## GitHub

- Has unlimited public repositories
- With unlimited collaborators
- Pay for private repositories
- However as a student or academic you can sign up for free private repositories
- GitHub is about  $\times 4$  more popular than Bitbucket

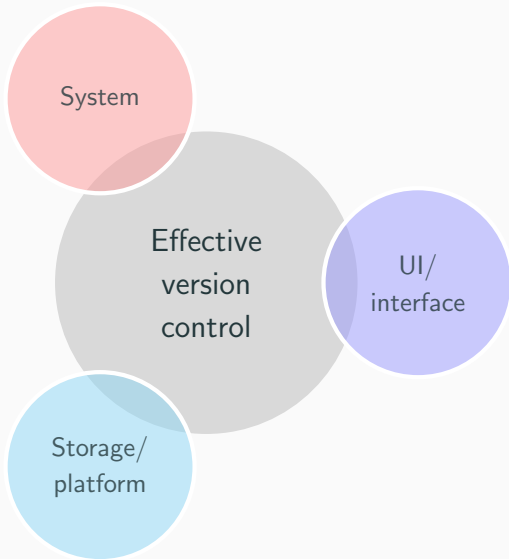


UI

---

# Which UI?

You can just use git with the terminal but there is a friendlier interface



# Which UI?

## The Terminal

You can just use the terminal with the raw git commands as seen before... but:



- If you are not using it anyway or are happy using it a graphical UI is nice
- You don't need to remember commands or repo names
- You need to install git for widows  
<https://git-scm.com/download/win>

```
C:\Windows\system32\cmd.exe

C:\Users\dgai\My Documents\GitHub\Presentations\GIT>git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   Figs/git/Dropbox.png

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Danny_AC_Taguchi.pdf
        modified:   Danny_AC_Taguchi.tex

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

        Figs/git/gitdesktop.png
        Figs/git/terminal.png

C:\Users\dgai\My Documents\GitHub\Presentations\GIT>git commit -m "added image f
or dropbox"
```

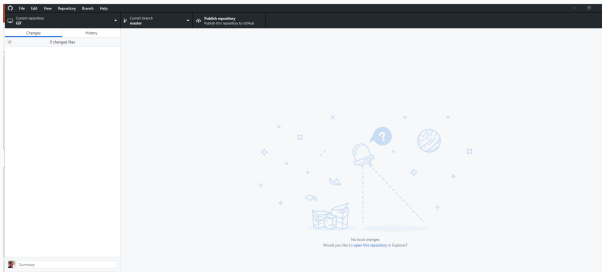
# Which UI?

## Git Desktop

The easiest to use and most popular UI is git desktop:

- This is made by GitHub and integrates very well with GitHub or local (on pc) repositories
- It will work with other hosting platforms but less easily
- Is the easiest to use!
- Will install git for windows

<https://git-scm.com/download/win>

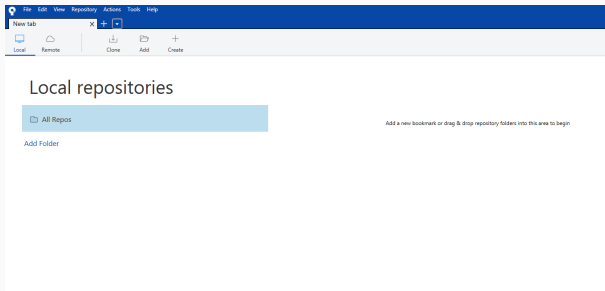


# Which UI?

## Sourcetree

- Sourcetree is owned by atlassian who owns Bitbucket
- It will work with other hosting platforms easily
- It has more complex controls
- Will install git for windows

<https://git-scm.com/download/win>



## Using git

---



# How to use git

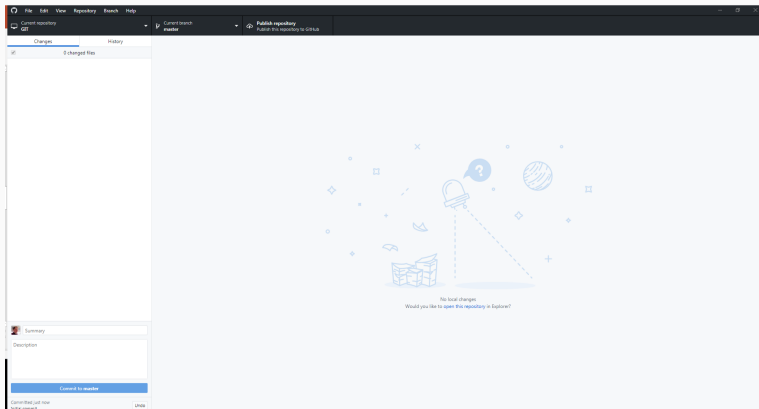
**This will assume use of GitHub and GitDesktop**

<https://desktop.github.com>

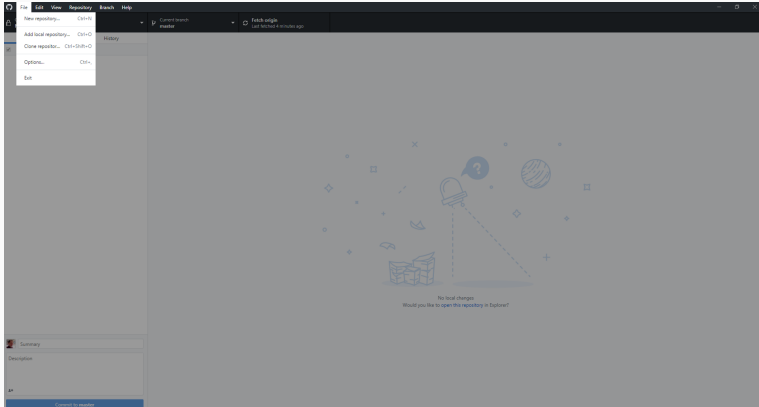
Following is a brief demonstration of the interface



# Adding a new repository

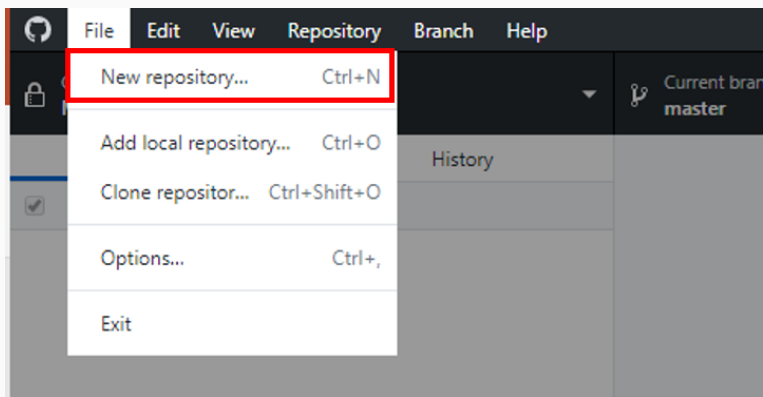


# Adding a new repository



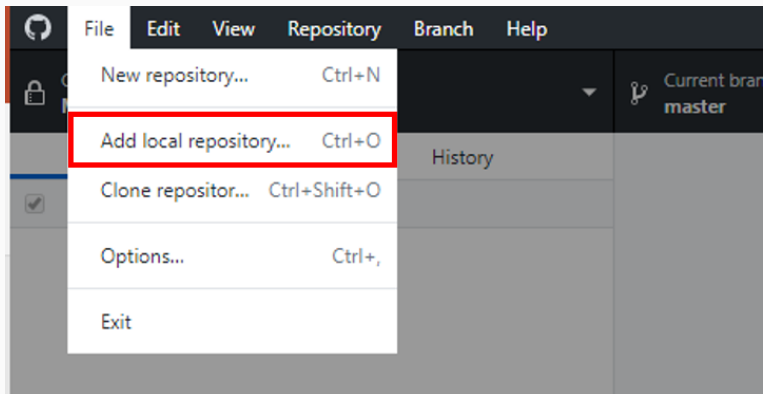
## Adding a new repository

This makes a totally new repository (with no '.git' folder)



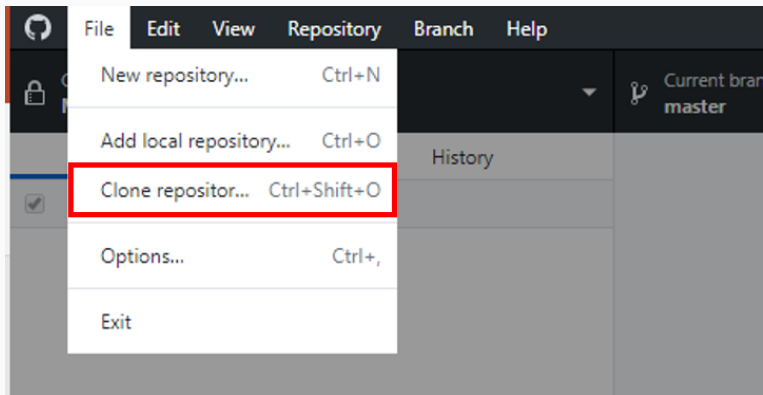
## Adding a new repository

adds an existing repository (with a '.git' folder)



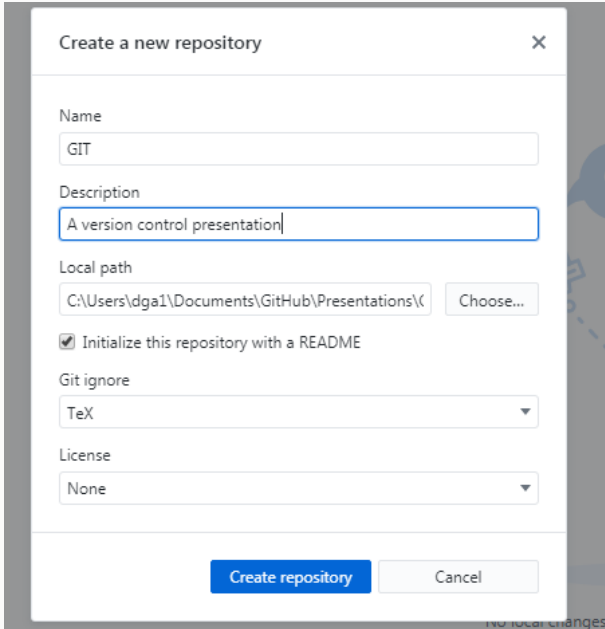
# Adding a new repository

Copies  
an online  
repositor-  
y to  
the PC



# Adding a new repository

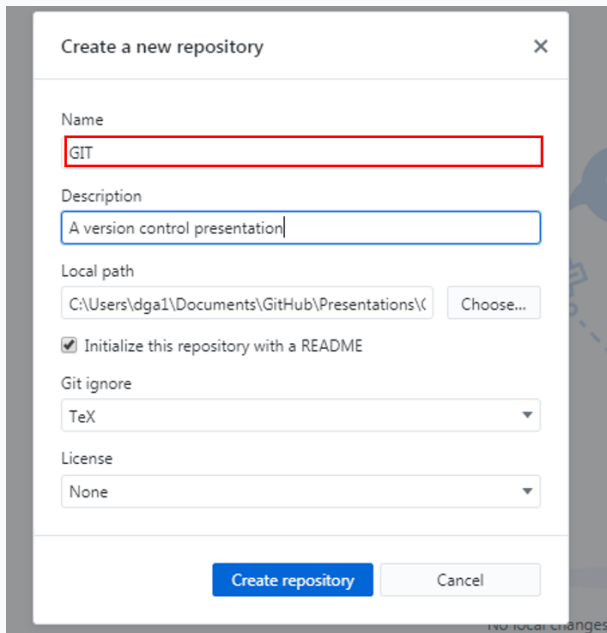
new repository  
menu



The screenshot shows a 'Create a new repository' dialog box with the following fields and options:

- Name:** A text input field containing 'GIT'.
- Description:** A text input field containing 'A version control presentation'.
- Local path:** A text input field containing 'C:\Users\dga1\Documents\GitHub\Presentations\(' and a 'Choose...' button to the right.
- Initialize this repository with a README:** A checked checkbox.
- Git ignore:** A dropdown menu showing 'TeX'.
- License:** A dropdown menu showing 'None'.
- Buttons:** 'Create repository' (blue) and 'Cancel' (grey) at the bottom.

## Adding a new repository

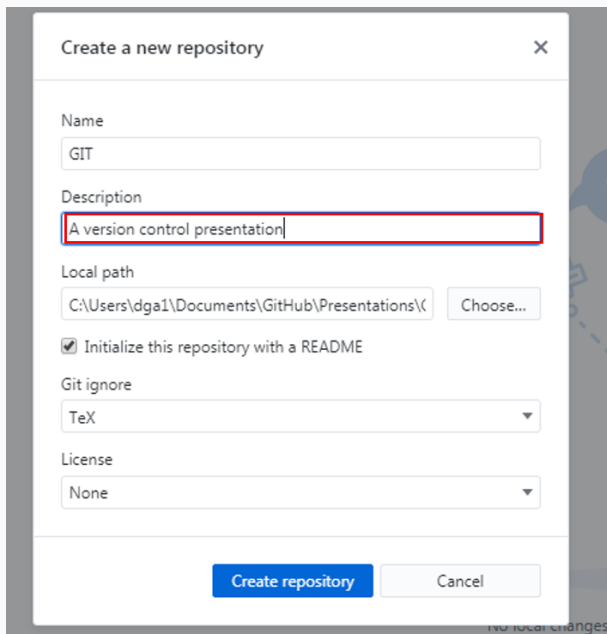


The screenshot shows a 'Create a new repository' dialog box with the following fields and options:

- Name:** A text input field containing 'GIT', highlighted with a red border.
- Description:** A text input field containing 'A version control presentation'.
- Local path:** A text input field containing 'C:\Users\dga1\Documents\GitHub\Ppresentations\(' and a 'Choose...' button.
- Initialize this repository with a README:** A checked checkbox.
- Git ignore:** A dropdown menu showing 'TeX'.
- License:** A dropdown menu showing 'None'.
- Buttons:** 'Create repository' (blue) and 'Cancel' (gray) at the bottom.



## Adding a new repository

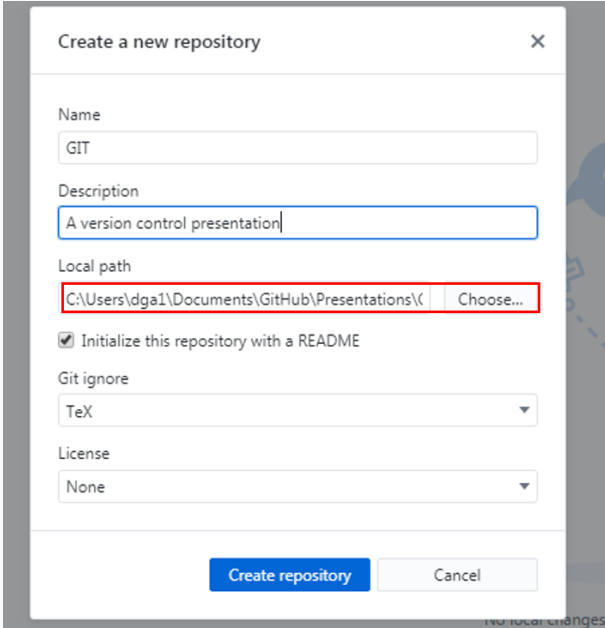


The screenshot shows a 'Create a new repository' dialog box with the following fields and options:

- Name:** A text input field containing 'GIT'.
- Description:** A text input field containing 'A version control presentation', which is highlighted with a red border.
- Local path:** A text input field containing 'C:\Users\dga1\Documents\GitHub\Ppresentations\(' and a 'Choose...' button to the right.
- Initialize this repository with a README:** A checked checkbox.
- Git ignore:** A dropdown menu showing 'TeX'.
- License:** A dropdown menu showing 'None'.
- Buttons:** 'Create repository' (blue) and 'Cancel' (grey) buttons at the bottom.

## Adding a new repository

The folder **in which** to store the git repository folder.



Create a new repository

Name  
GIT

Description  
A version control presentation

Local path  
C:\Users\dga1\Documents\GitHub\Presentations\ Choose...

☒ Initialize this repository with a README

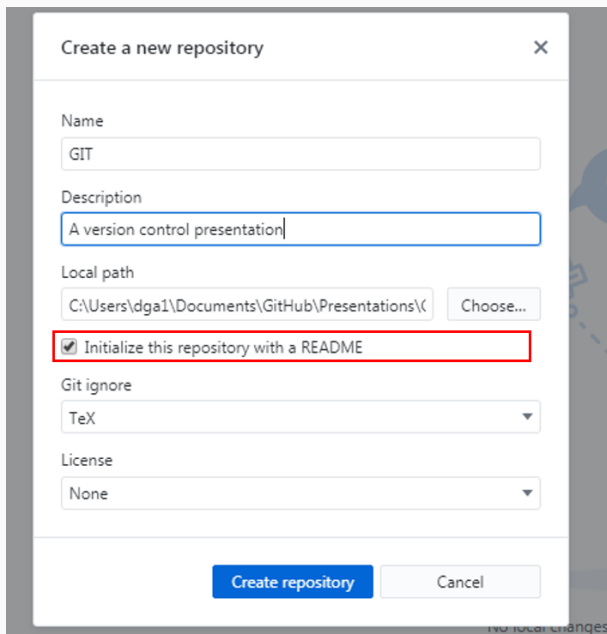
Git ignore  
TeX

License  
None

Create repository Cancel

## Adding a new repository

This will initialise the repository with a readme containing the description.



**Create a new repository** [X]

Name  
GIT

Description  
A version control presentation

Local path  
C:\Users\dga1\Documents\GitHub\Presentations\ [Choose...]

☒ Initialize this repository with a README

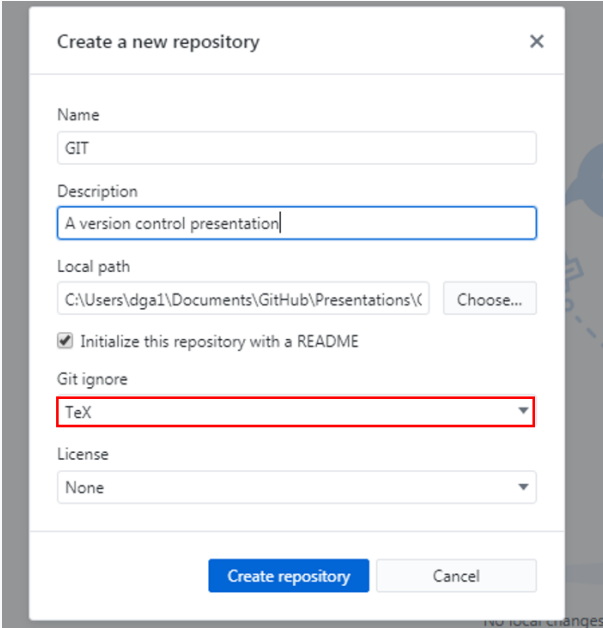
Git ignore  
TeX

License  
None

**Create repository** Cancel

# Adding a new repository

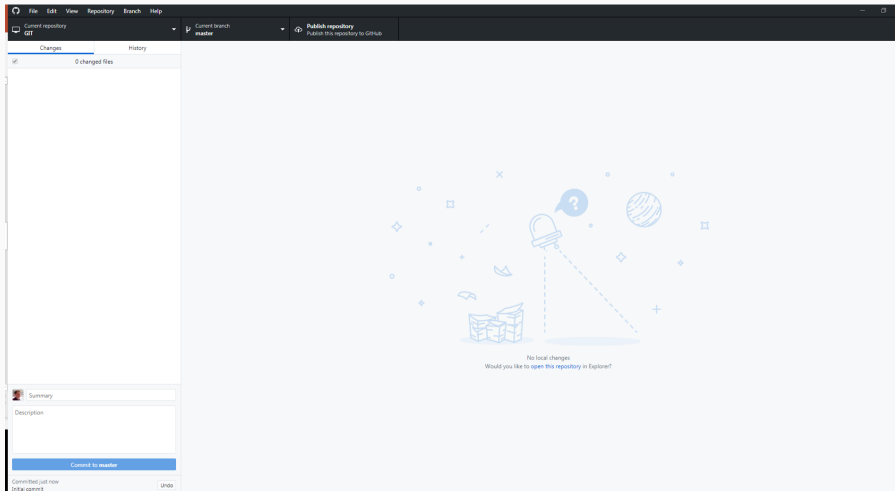
This is a file called  
'`.gitignore`'  
which lists files  
& folders to not  
version control



The screenshot shows a 'Create a new repository' dialog box with the following fields and options:

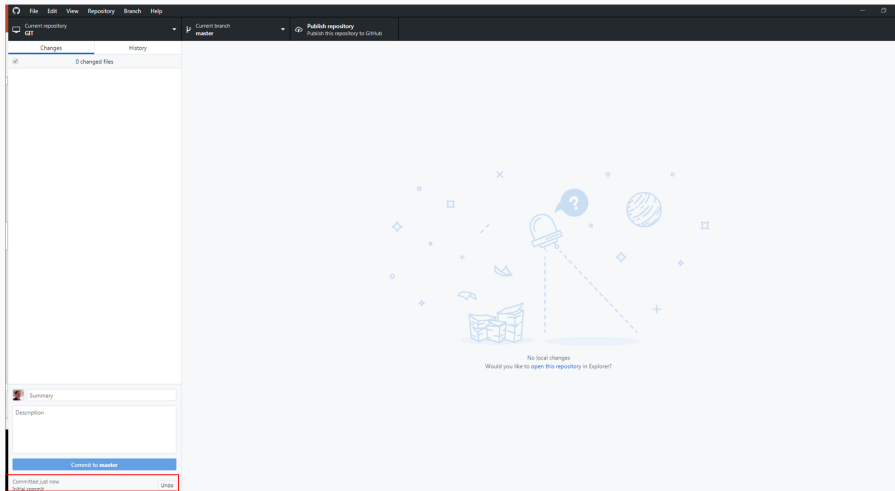
- Name:** A text input field containing 'GIT'.
- Description:** A text input field containing 'A version control presentation'.
- Local path:** A text input field containing 'C:\Users\dga1\Documents\GitHub\Ppresentations\' and a 'Choose...' button.
- Initialize this repository with a README:** A checked checkbox.
- Git ignore:** A dropdown menu with 'TeX' selected and highlighted by a red rectangle.
- License:** A dropdown menu with 'None' selected.
- Buttons:** 'Create repository' (blue) and 'Cancel' (grey) buttons at the bottom.

# Using the repository



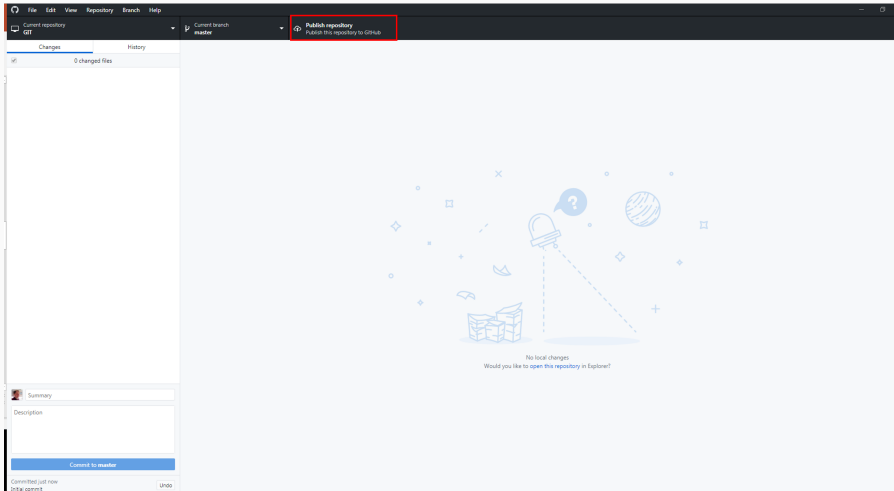
# Using the repository

Now we have an initial commit of the readme done



# Using the repository

We can carry on making local commits or publish the repository to GitHub



# Using the repository

We can publish to GitHub

Publish repository

GitHub.com

Enterprise

Name

GIT

Description

A version control presentation

☒ Keep this code private

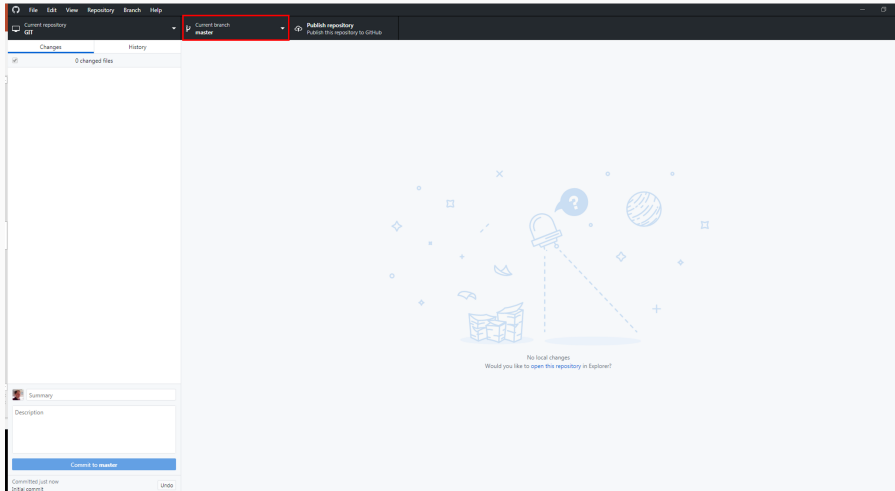
Publish repository

Cancel



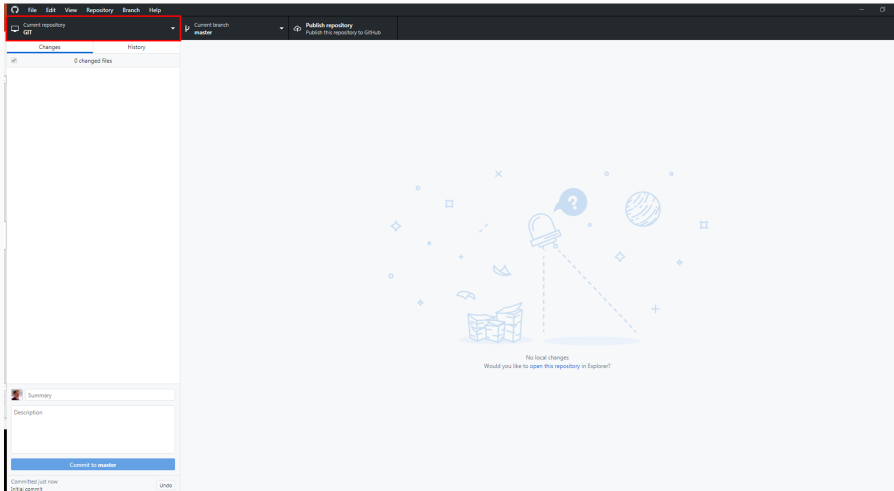
# Using the repository

## Make branches



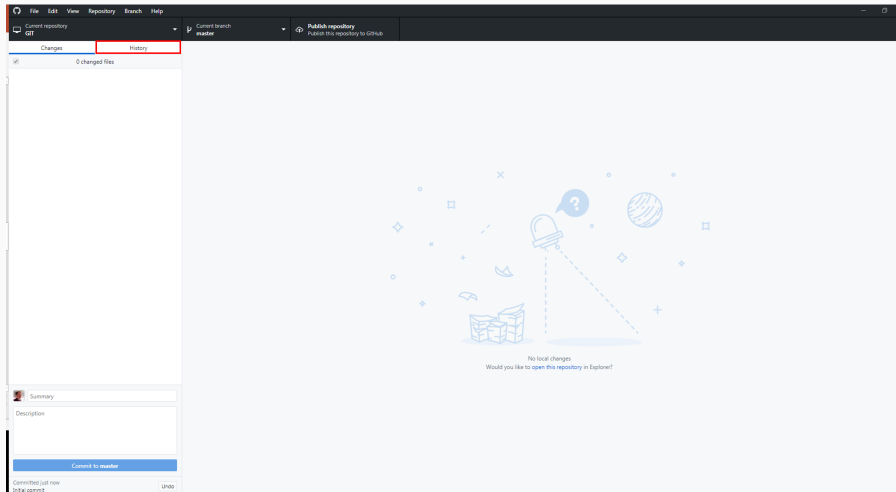
# Using the repository

## Switch to another repository



# Using the repository

Look at the history of this repository



# Using the repository

## Look at the history of this repository

The screenshot displays the GitHub web interface for a repository. The top navigation bar shows the current repository is 'GIT' and the current branch is 'master'. A 'Push origin' button indicates the last fetch was 2 minutes ago. The left sidebar contains a list of commits, with the most recent one highlighted: 'A figure for committing changes on GHD' by Danny George Awty-Carroll [dga1], committed just now. Below this, two other commits are listed: 'figures for Git Hub Desktop' (committed 8 minutes ago) and 'Initial commit' (committed 19 minutes ago). The main content area shows the commit details for the selected commit, including the commit message 'A figure for committing changes on GHD', the committer's name and email, the commit hash 'ddba7d0', and a note that 1 file was changed. The file 'Figs\GHD\outline\_08.png' is listed as the changed file. On the right, a diff view is shown, displaying the changes to the file 'Figs\GHD\outline\_08.png'. The diff view includes a menu bar with 'File', 'Edit', 'View', 'Repository', 'Branch', and 'Help'. The 'Changes' tab is selected, showing a list of changed files with checkboxes and a green plus icon for each file. The file 'Figs\GHD\outline\_08.png' is listed as the changed file.

Current repository: GIT

Current branch: master

Push origin: Last fetched 2 minutes ago

Changes | History

A figure for committing changes on GHD

Danny George Awty-Carroll [dga1] committed just now

figures for Git Hub Desktop

Danny George Awty-Carroll [dga1] committed 8 minutes ago

Initial commit

Danny George Awty-Carroll [dga1] committed 19 minutes ago

A figure for committing changes on GHD

Danny George Awty-Carroll [dga1] committed ddba7d0 1 changed file

Figs\GHD\outline\_08.png

File Edit View Repository Branch Help

Current repository: GIT

Current branch: master

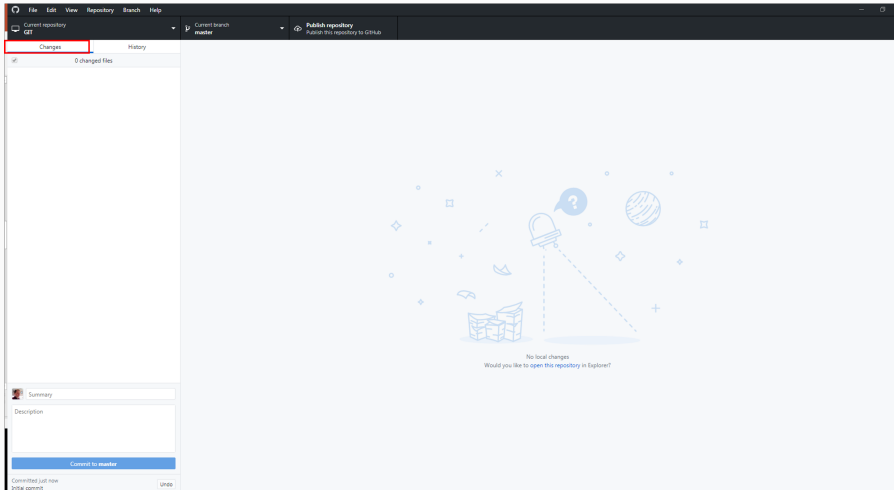
Changes | History

1 changed file

Figs\GHD\outline\_08.png

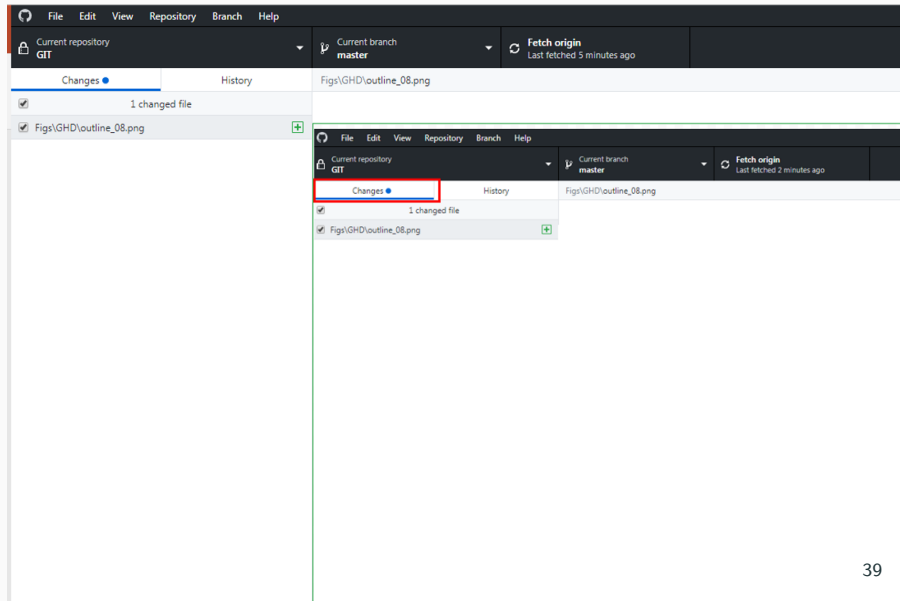
# Using the repository

Or look at how the files have changed from the records (same as *git status*)



# Using the repository

When there are changes you can make a new commit



# Using the repository

## An example of some small changes to this presentation

Current repository  
GIT

Current branch  
master

Fetch origin  
Last fetched a minute ago

Changes

History

2 changed files

Danny\_AC\_git.pdf

Danny\_AC\_git.tex

An example of some small changes to this presentation

Could add a longer description

Add co-authors

Commit to master

Danny\_AC\_git.tex

```
@@ -27,7 +27,7 @@
\maketitle

-
+ \setbeamertemplate{frame footer}[ ]
\begin{frame}[Sections]
This will focus on using git in windows with a UI
\newline
@@ -72,7 +72,7 @@ Some of this is solved in programs like word with track changes (so you see who
\begin{frame}[fragile][what version control systems can do]
\begin{itemize}
\item Give line by line user by user line changes
+ \item Give line by line changes linked to a user (which are kept forever)
\item Makes a stretcher to the version control so there can be side branches (this is where the project can take a detour and be merged back in later)
\item Minimise problems of multiple users working on the same document at the same time
\end{itemize}
@@ -185,7 +185,7 @@ In the terminal interface there are some useful commands:
\end{frame}
}

-
+ \setbeamertemplate{frame footer}[ ]
\section{Platform}

@@ -260,8 +260,8 @@ You can just use git with the terminal but there is a friendlier interface\
System, Storage/\platform, UI/\interface
\end{column}
\begin{column}[.2\textwidth]
\includegraphics[width=2cm]{figs/git/terminal} \newline \newline \newline
\includegraphics[width=2cm]{figs/git/gitdesktop} \newline \newline \newline
+ \includegraphics[width=2cm]{figs/git/terminal} \newline \newline \newline
+ \includegraphics[width=2cm]{figs/git/gitdesktop} \newline \newline \newline
\includegraphics[width=2cm]{figs/git/Sourcetree}
\end{column}
\end{columns}
@@ -494,28 +494,28 @@ Following is a brief demonstration of the interface
```

## Questions & Demo

---



|   | COMMENT                            | DATE         |
|---|------------------------------------|--------------|
| ○ | CREATED MAIN LOOP & TIMING CONTROL | 14 HOURS AGO |
| ○ | ENABLED CONFIG FILE PARSING        | 9 HOURS AGO  |
| ○ | MISC BUGFIXES                      | 5 HOURS AGO  |
| ○ | CODE ADDITIONS/EDITS               | 4 HOURS AGO  |
| ○ | MORE CODE                          | 4 HOURS AGO  |
| ○ | HERE HAVE CODE                     | 4 HOURS AGO  |
| ○ | AAAAAAAAA                          | 3 HOURS AGO  |
| ○ | ADKFJSLKDFJSDKLFJ                  | 3 HOURS AGO  |
| ○ | MY HANDS ARE TYPING WORDS          | 2 HOURS AGO  |
| ○ | HAAAAAAAAAANDS                     | 2 HOURS AGO  |

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.