1. Preparation

Open the Git prompt, go to the directory that you want to version control, or make a new directory. Here I would like to use Git to version control my PhD thesis. The files are in a folder named **thesis**.

```
$cd Documents
rrd09@XXXXX ~/Documents
$cd thesis
rrd09@XXXXX ~/Documents/thesis
```

2. Initialise Git

Just type **git init**.

```
$ git init
Initialized empty Git repository in c:/Users/rrd09/Documents/thesis/.git/
```

3. Add some files

If you already have some files, you can start adding them using **git add [file name]**. Here I add some of my Latex files.

```
rrd09@XXXXX ~/Documents/thesis (master)
$ git add dissertation.tex
```

You may see this warning message: **warning: LF will be replaced by CRLF in dissertation.tex. The file will have its original line endings in your working directory**, which just tells you that Git will unify the "end of line" representations in Unix (LF) and Windows (CRLF), nothing to worry about.

Git supports wildcards too:

```
$ git add *.tex
```

and you can also add folders, denoted by the tailing forward slash **/**:

```
$ git add introduction/
$ git add background*/
```

3. Once you finished adding files, commit the change:

```
$ git commit –m 'Initial version of thesis'
[master (root-commit) 492a8b4] Initial version of thsis
 48 files changed, 11944 insertions(+)
 create mode 100644 dissertation.tex
 ...
 create mode background_fs/001.aux
 create mode background_fs/001.tex
 create mode background_nim/001.aux
 create mode background_nim/001.tex
 ...
 create mode 100644 introduction/001.aux
 create mode 100644 introduction/001.texx
 ...
```

Note that you must provide a (hopefully meaningful) commit message using **–m '[message]'**, which will help you to keep track of these changes in the future.

4. Check the status of your repository

```
$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       abstract.aux
#       dissertation.aux
#       ...
#       dissertation.synctex.gz
#       ...
```

Here I am intentionally not adding some files as they are intermediate outputs of the Latex editor. Similarly, you may not want to add your complier outputs, temporary files, etc.

5. Un-track some files

There are times when you accidentally added some files, like **background_fs/001.aux**, you may un-track them by doing **git rm --cached [file name]**.

```
$ git rm --cached back*/*.aux
rm 'background_fs/001.aux'
rm 'background_nim/001.aux'
```

and lets commit these changes:

```
$ git commit –m 'Removed tracking on some unnecessary files'
[master fc9ce44] Removed tracking on some unnecessary files'
 12 files changed, 861 deletions(-)
 ...
 delete mode 100644 background_fs/001.aux
 delete mode 100644 background_nim/001.aux
 ...
```

6. Ignoring files

To stop Git from worrying about these files all together, we can add them to the ignore list. For this, you need to make a new file named **.gitignore**, and populate it with some patterns, or file names. For example:

```
$ echo "*.aux" > .gitignore
$ echo "*.gz" >> .gitignore
```

will add any file matching the **.aux** extension to the ignore list, regardless of where they are in the entire repository. Regular expressions such as ***.[oa]**, matching any file ending in **.o** or **.a**, and ***~**, matching anything ending in **~**, are all accepted.

After adding some files in **.gitignore**, we have:

```
$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       .gitignore
nothing added to commit but untracked files present (use "git add" to track)
```

The ignore list can/should of course, also be added to the repository.