

# Version Control with Git



Ben R. Fitzpatrick

PhD Candidate, Statistical Science, Mathematical Sciences School, Queensland University of Technology

0000-0003-1916-0939

[github.com/brfitzpatrick/](https://github.com/brfitzpatrick/)

@benrfitzpatrick

# Download Git for your Operating System

<http://git-scm.com/downloads>



# git --fast-version-control

**About**

**Documentation**

**Blog**

**Downloads**

GUI Clients

Logos

**Community**

## Downloads



Mac OS X



Windows



Linux



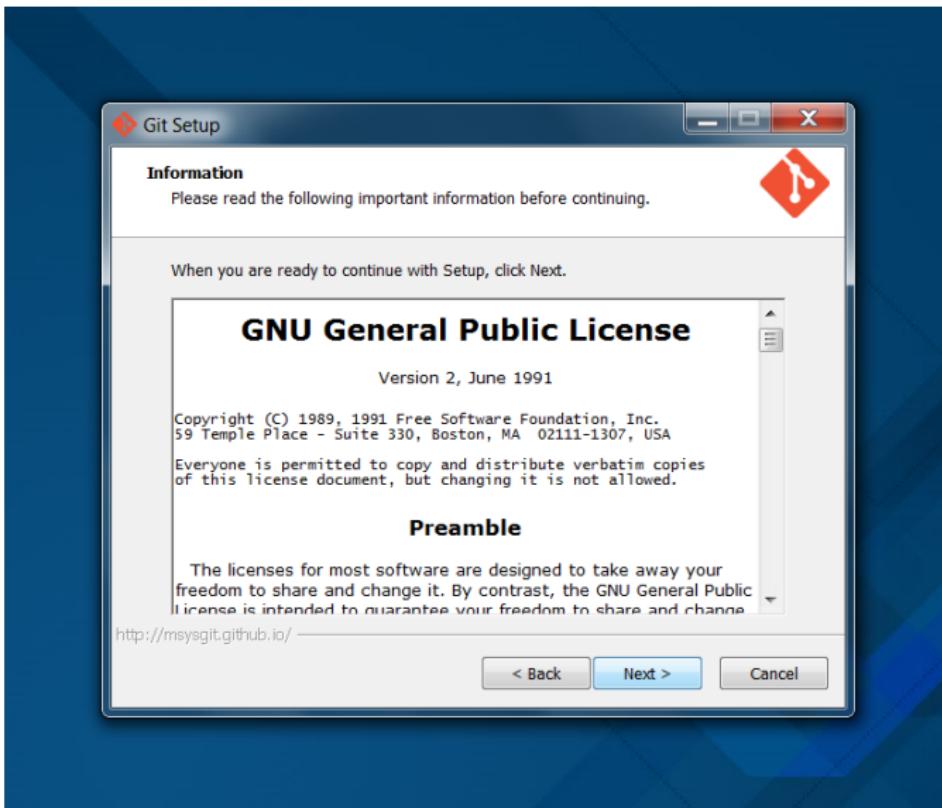
Solaris

Older releases are available and the [Git source repository](#) is on GitHub.

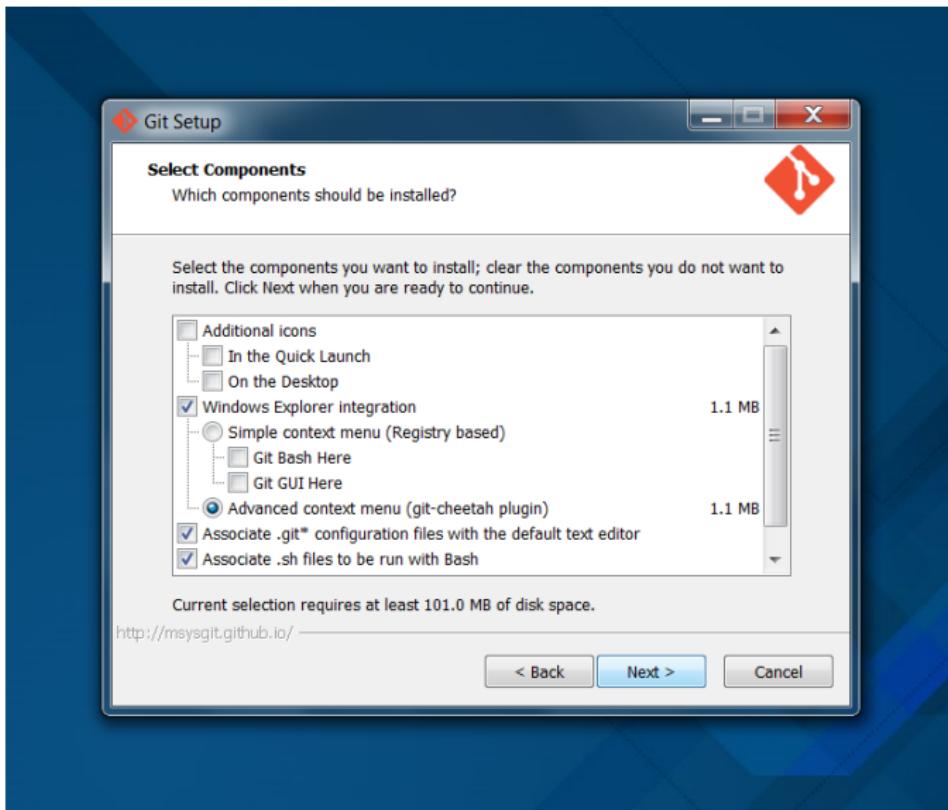
# Installing Git on MS Windows: Step 1



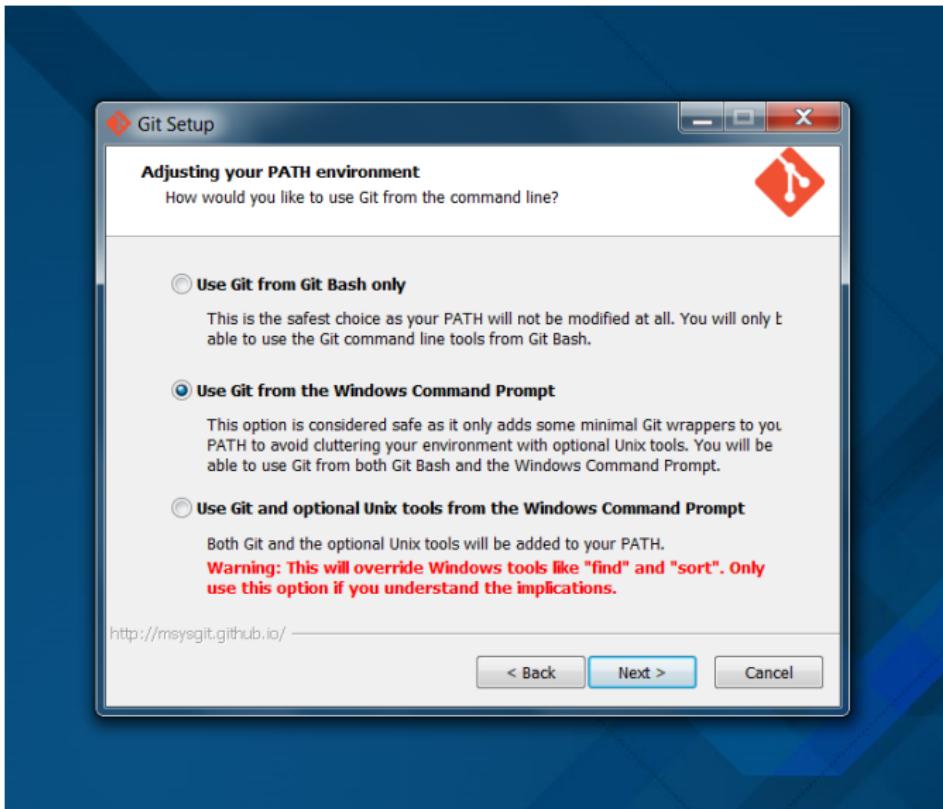
# Installing Git on MS Windows: Step 2



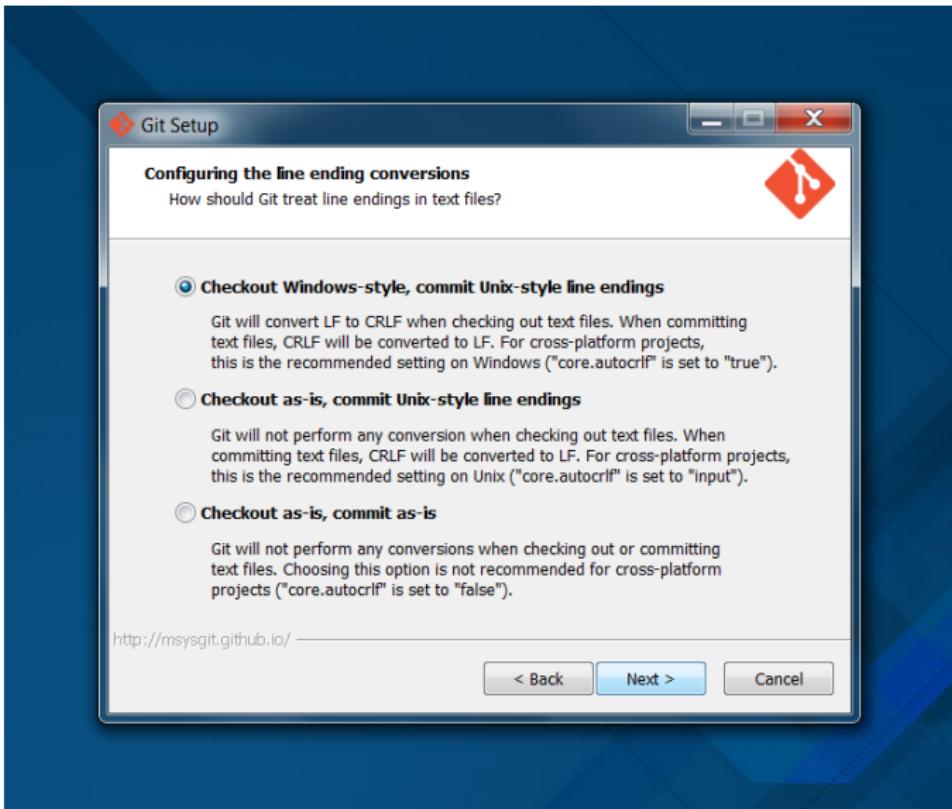
# Installing Git on MS Windows: Step 3



# Installing Git on MS Windows: Step 4

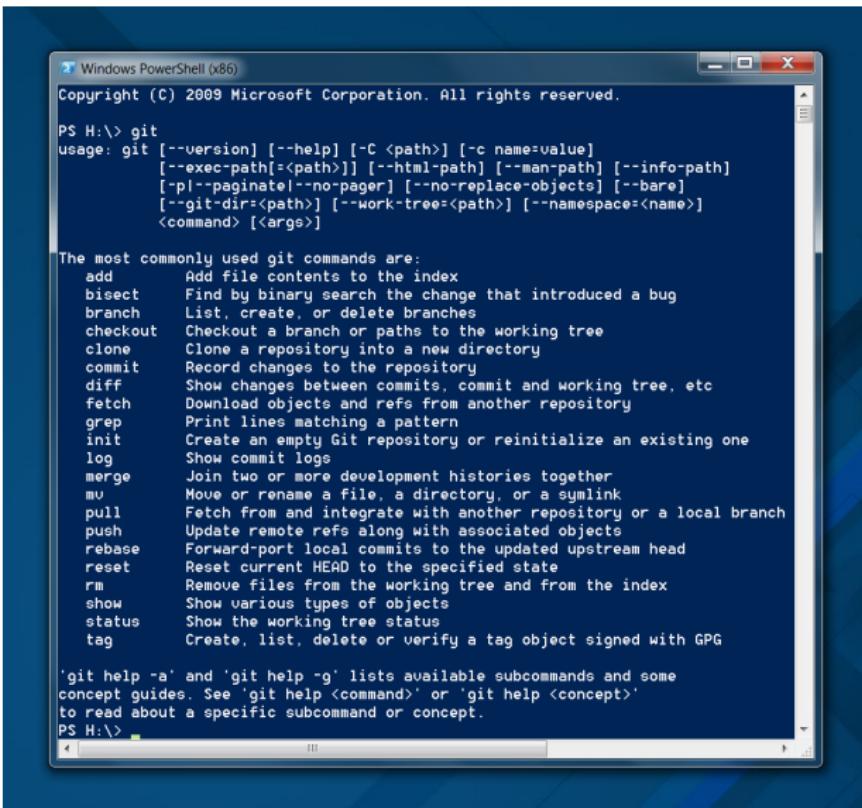


# Installing Git on MS Windows: Step 5



# Check Git is installed on MS Windows

Open a PowerShell and type *git*



A screenshot of a Windows PowerShell window titled "Windows PowerShell (x86)". The window displays the output of the command "git". The output includes the copyright notice "Copyright (C) 2009 Microsoft Corporation. All rights reserved.", usage information for the "git" command, a list of commonly used git commands with their descriptions, and a note about available subcommands and concept guides.

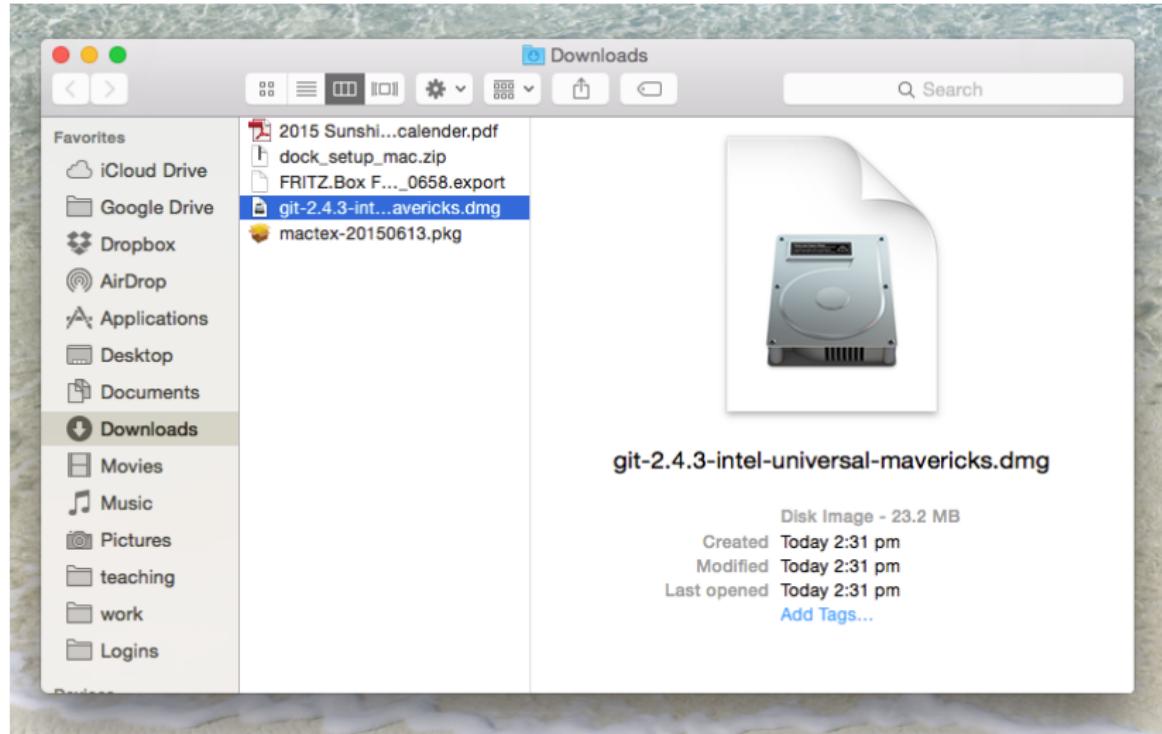
```
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS H:\> git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

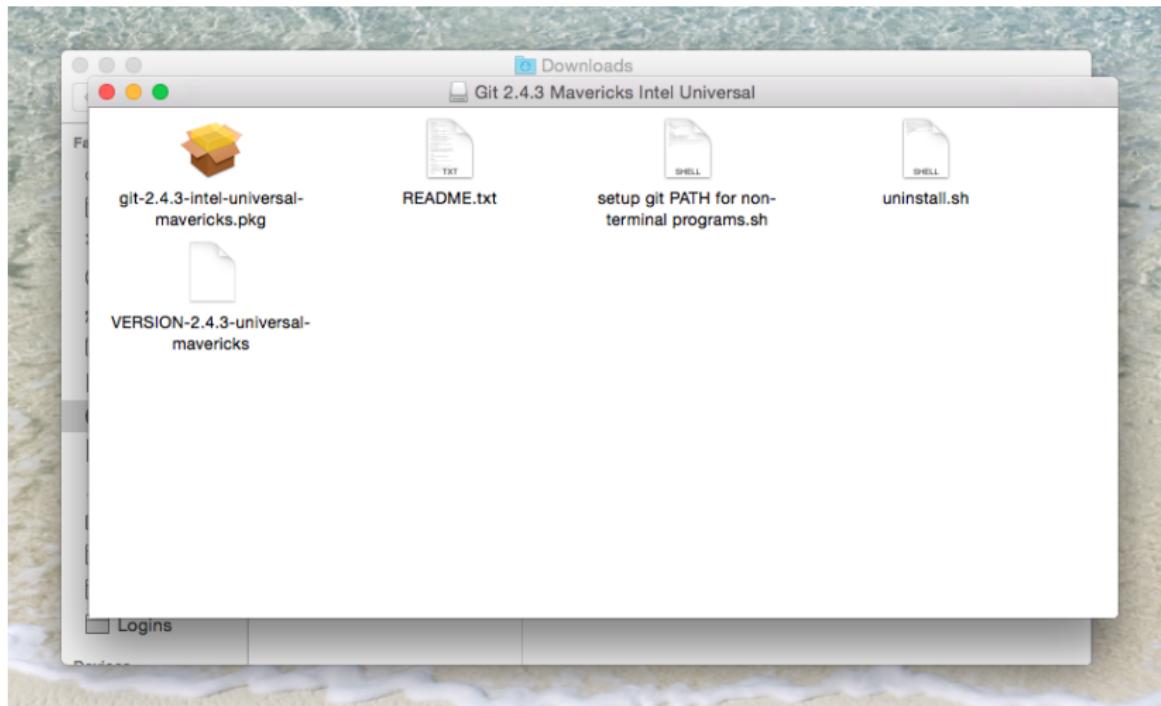
The most commonly used git commands are:
  add      Add file contents to the index
  bisect   Find by binary search the change that introduced a bug
  branch   List, create, or delete branches
  checkout Checkout a branch or paths to the working tree
  clone    Clone a repository into a new directory
  commit   Record changes to the repository
  diff     Show changes between commits, commit and working tree, etc
  fetch   Download objects and refs from another repository
  grep    Print lines matching a pattern
  init    Create an empty Git repository or reinitialize an existing one
  log     Show commit logs
  merge   Join two or more development histories together
  mv     Move or rename a file, a directory, or a symlink
  pull    Fetch from and integrate with another repository or a local branch
  push    Update remote refs along with associated objects
  rebase  Forward-port local commits to the updated upstream head
  reset   Reset current HEAD to the specified state
  rm     Remove files from the working tree and from the index
  show    Show various types of objects
  status  Show the working tree status
  tag    Create, list, delete or verify a tag object signed with GPG

'git help -a' and 'git help -g' lists available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
PS H:\>
```

# Installing Git on MacsOS: Step 1



# Installing Git on MacsOS: Step 2



# Installing Git on MacsOS: Step 3



**"git-2.4.3-intel-universal-mavericks.pkg"**  
can't be opened because it is from an  
unidentified developer.

Your security preferences allow installation of only  
apps from the Mac App Store and identified  
developers.

"git-2.4.3-intel-universal-mavericks.pkg" is on the  
disk image "git-2.4.3-intel-universal-mavericks.dmg".  
Google Chrome downloaded this disk image today at  
2:31 pm from sourceforge.net.

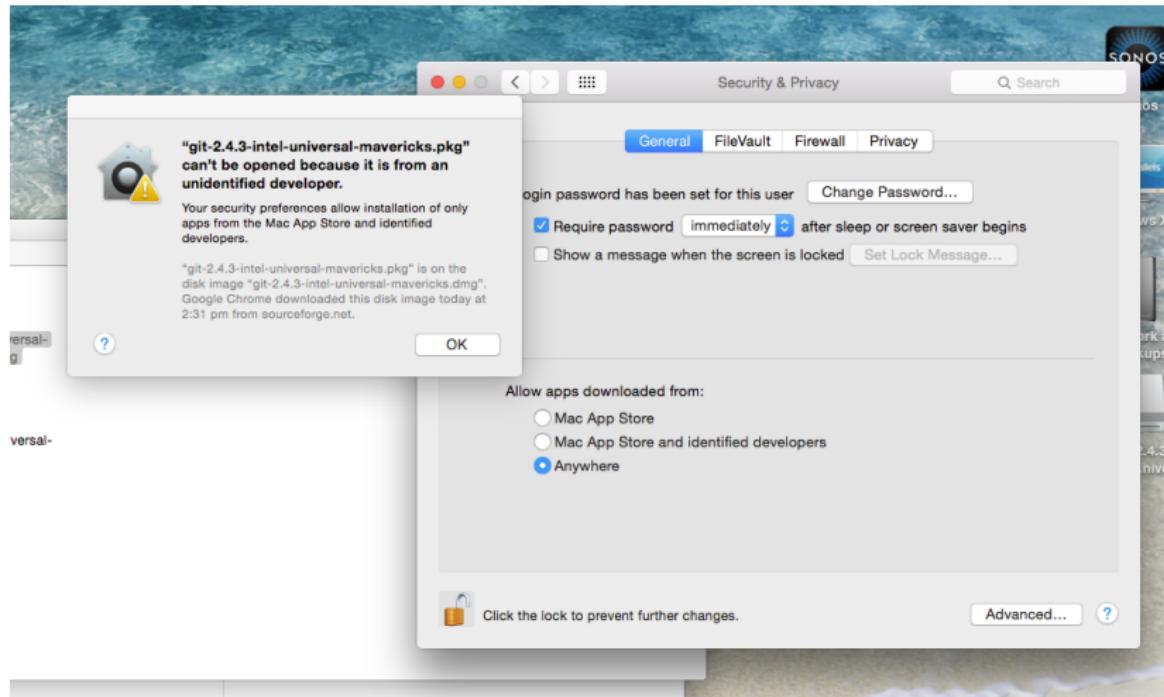


OK

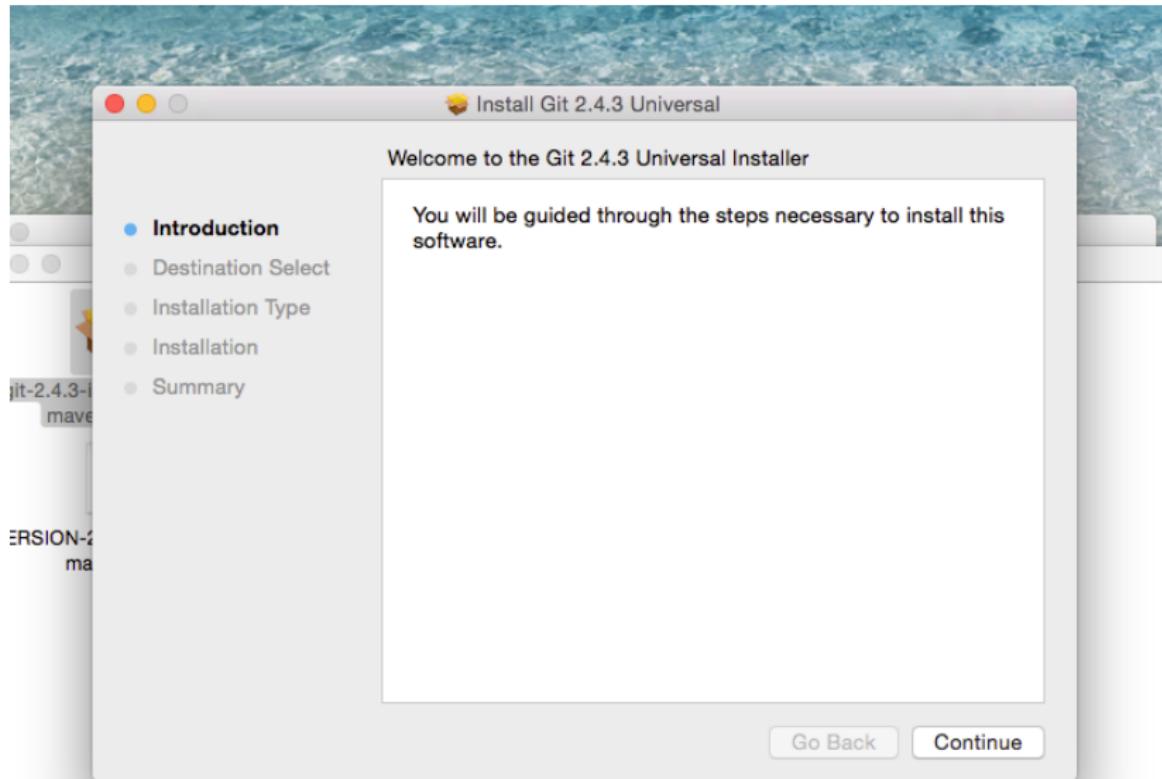
# Installing Git on MacsOS: Step 4

You may need to relax your security setting briefly

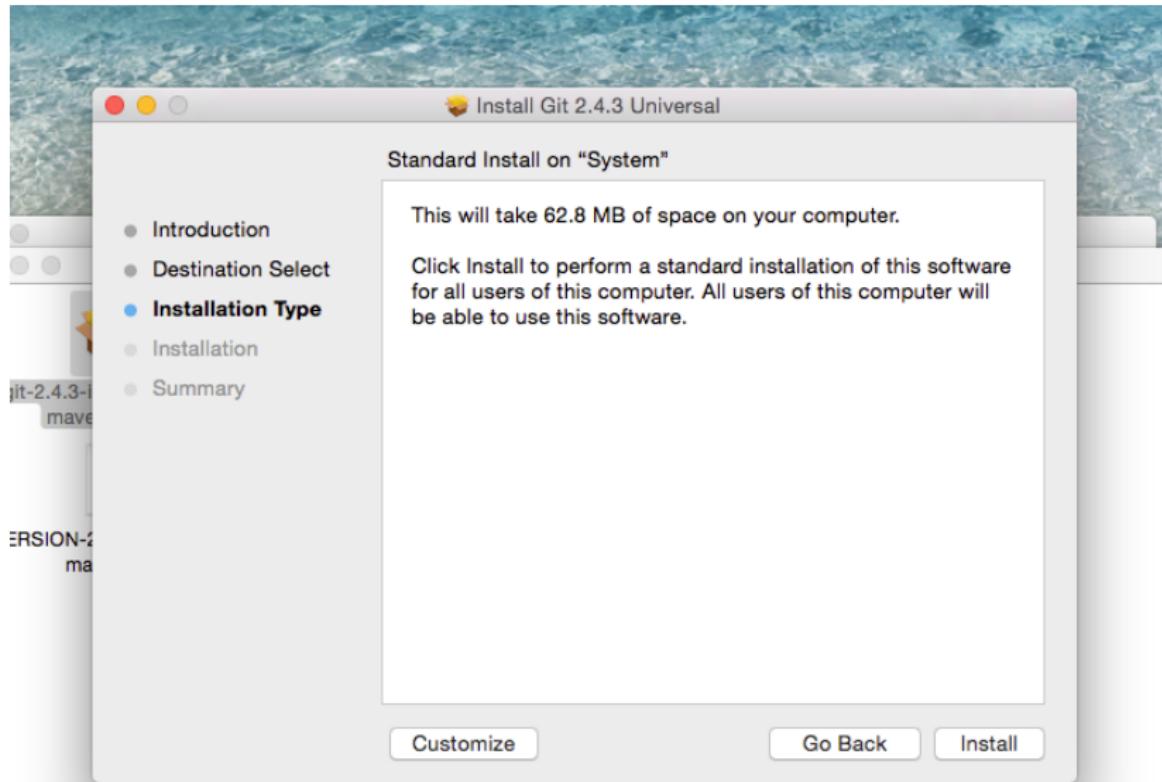
If this makes you uncomfortable you're welcome to watch the session rather than participating



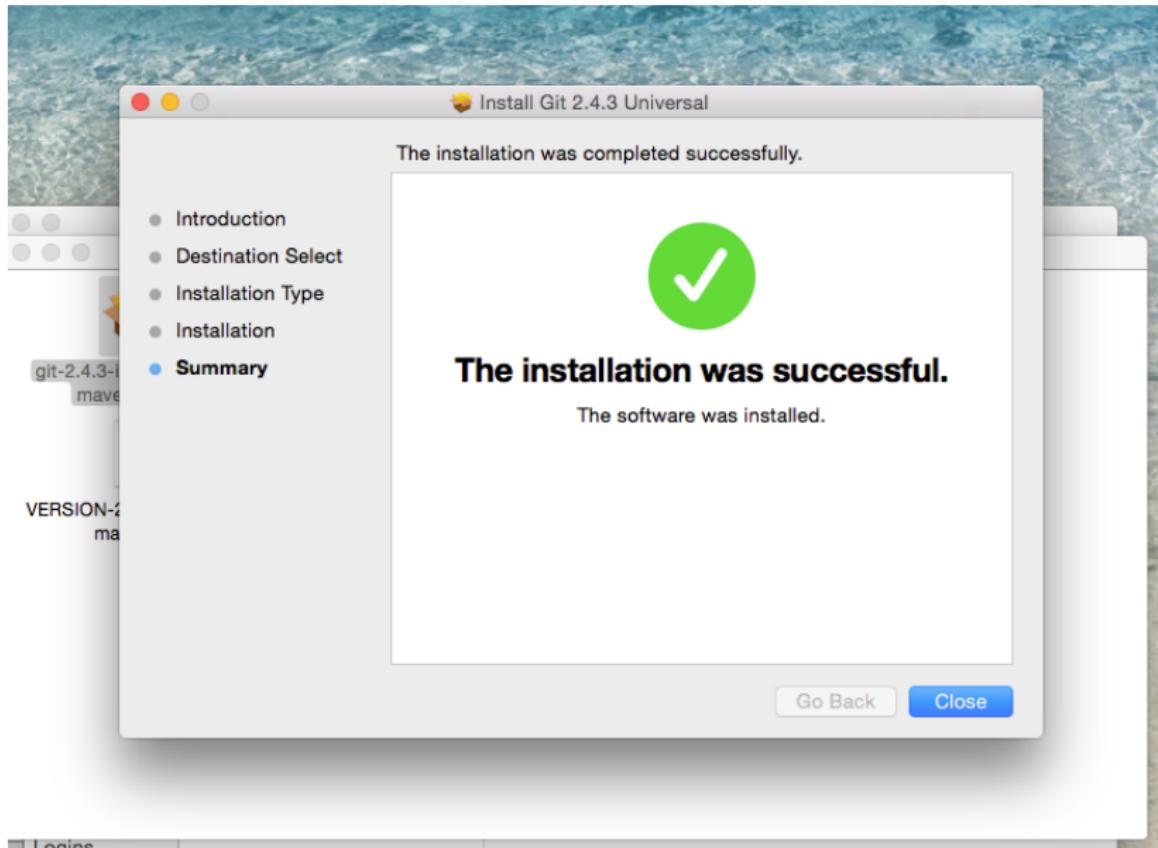
# Installing Git on MacsOS: Step 5



# Installing Git on MacsOS: Step 6

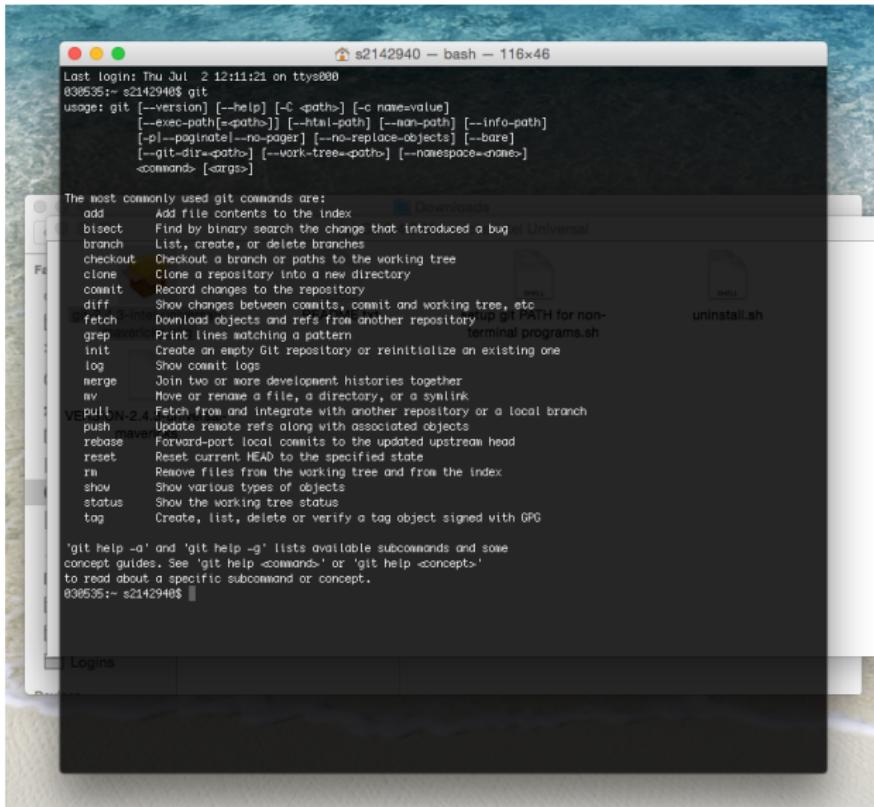


# Installing Git on MacsOS: Step 7



# Installing Git on MacsOS: Step 8

Check Git is installed:



```
s2142940 - bash - 116x46
Last login: Thu Jul  2 12:11:21 on ttys000
030535:~ s2142940$ git
usage: git [--version] [--help] [-c <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

The most commonly used git commands are:
add      Add file contents to the index
bisect   Find by binary search the change that introduced a bug
branch   List, create, or delete branches
checkout  Checkout a branch or paths to the working tree
clone    Clone a repository into a new directory
commit   Record changes to the repository
diff     Show changes between commits, commit and working tree, etc
fetch   Download objects and refs from another repository [ip git PATH for non-
grep    Print lines matching a pattern
init     Create an empty Git repository or reinitialize an existing one
log      Show commit logs
merge   Join two or more development histories together
mv      Move or rename a file, a directory, or a symlink
pull   Fetch from and integrate with another repository or a local branch
push   Push remote refs along with associated objects
release Forward-port local commits to the updated upstream head
reset   Reset current HEAD to the specified state
rm      Remove files from the working tree and from the index
show    Show various types of objects
status   Show the working tree status
tag     Create, list, delete or verify a tag object signed with GPG

'git help -a' and 'git help -g' lists available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
030535:~ s2142940$
```

# Version Control Software

Have you ever put numbers or dates at the end of a file name to keep different versions of it?

- If yes then you have already used version control!

Have you ever used track changes and comments to take turns editing a MS Word document with one or more others?

- If yes then you have collaboratively edited a file!

Version Control Software:

- formalizes these concepts
- can do a lot of the organisational work for you
- makes reverting to previous versions without losing work easy

# Version Control when Working Alone

## Version Control for Solo Authoring File

- You essentially have a time stamped undo button which will allow you to go review or revert to previous versions of the file
- you can even make multiple child versions fo the same file say for different audiences or to trial different orders and styles in which to write a report or program

## Version Control when Collaborating

Greatly simplifies your life Control over which changes different contributers make get added to the definitive version of a file support for file to split into two child copies which can then be edited towards very different end points but still having the option to systematically recombine them at any stage

# Why Git?

Git is free and open source

Git is available for most operating systems

Git is Distributed and Fast

Two major code hosting services (GitHub & BitBucket) support version control with Git and both services have free account options

# Git is Distributed Version Control

Every contributor has a copy of the entire database ( all revisions of all files) two contributors can both work on the files in their copy of the database independently then their combined modifications can be merged systematically

# Why Git on the Command Line?

Makes discrete components of the workflow obvious this in turn will enable you to migrate easily to one of the many GUIs

The workflows we will practise today will work exactly the same way on MS Windows, MacOS and GNU+Linux

We have now been using a command line interface for 1.5 days to interact with the R program it's time to take it to the next level

# Plan for this Module - Part 1

## Solo Version Control

- Set up Git on your laptops to communicate with the GitHub servers
- Create a local 'clone' of a remote repository
- Make some changes to your local copy of the files
- commit these changes to the git version control system
- push these changes to your repository
- make some more changes
- revert to a previous version of the file

# Plan for this Module - Part 2

## Collaborating with Git & GitHub

- fork
- branch
- edit
- commit
- pull request
- merge

# Creating a Repository on the GitHub Servers

## Log into the GitHub Web Service

Search GitHub

Pull requests Issues Gist

Contribution Repositories Public activity

Edit profile

New

Find a repository Search All Public Private Sources Forks Merges

**Intro to R**  
Materials for an Introductory Short Course on the R Language and Environment for Statistical Computing and Graphics.  
Updated 2 hours ago

R ★ 1 0

**fork\_for\_ben**  
forked from sk8auer/fork\_for\_ben  
Updated 2 days ago

R ★ 0 0

**fork\_this\_repo**  
practise collaborating with GitHub  
Updated 4 days ago

R ★ 0 3

0 Followers 0 Starred 0 Following

# Creating a Repository on the GitHub Servers

## Complete the Details

Search GitHub Pull requests Issues Gist

Owner Repository name  
 brfitzpatrick / my\_new\_repo ✓

Great repository names are short and memorable. Need inspiration? How about [scaling-octo-ironman](#).

Description (optional)

 Public Anyone can see this repository. You choose who can commit.

 Private You choose who can see and commit to this repository.

Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: [GNU General Public License v3.0](#) ⓘ

**Create repository**

# Creating a Repository on the GitHub Servers

## View your New Repository

The screenshot shows a GitHub repository page for the user 'brfritzpatrick' named 'my\_new\_repo'. The top navigation bar includes links for 'This repository', 'Search', 'Pull requests', 'Issues', and 'Gist'. The repository name 'brfritzpatrick / my\_new\_repo' is displayed prominently. On the right side, there are buttons for 'Unwatch', 'Star', 'Fork', and a dropdown menu. Below the repository name, there are fields for 'Description' (containing 'Short description of this repository') and 'Website' (containing 'Website for this repository (optional)'). A 'Save' button and a 'Cancel' link are also present. To the right of these fields, there are summary statistics: 1 commit, 1 branch, 0 releases, and 1 contributor. The main content area shows the repository's history with a single commit from 'brfritzpatrick' authored just now. The commit message is 'Initial commit'. The commit hash is '4930beee35'. Below the commit, there are links for 'LICENSE' and 'README.md', both of which show 'Initial commit' status. A preview of the 'README.md' file content is shown, featuring the text 'my\_new\_repo'. On the right sidebar, there are links for 'Code', 'Issues' (0), 'Pull requests' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'. At the bottom, there is an 'SSH clone URL' field containing 'git@github.com:brf...', a note about cloning with HTTPS, SSH, or Subversion, and a 'Download ZIP' button.

This repository

Search

Pull requests Issues Gist

brfritzpatrick / my\_new\_repo

Unwatch 1 Star 0 Fork 0

Description

Short description of this repository

Website

Website for this repository (optional)

Save or Cancel

1 commit 1 branch 0 releases 1 contributor

branch: master my\_new\_repo / +

Initial commit

brfritzpatrick authored just now latest commit 4930beee35

LICENSE Initial commit just now

README.md Initial commit just now

README.md

my\_new\_repo

Code

Issues 0

Pull requests 0

Wiki

Pulse

Graphs

Settings

SSH clone URL

git@github.com:brf...

You can clone with HTTPS, SSH, or Subversion.

Download ZIP

# Cloning your new Repository to your Hard Drive

First, create a folder on your Hard Drive in which to store your Git Repositories.

We are now going to use the Git command line application to 'clone' your new repository from the GitHub server to your hard drive.

Git is distributed version control so a Git 'clone' is a complete copy of the entity of the repository i.e.

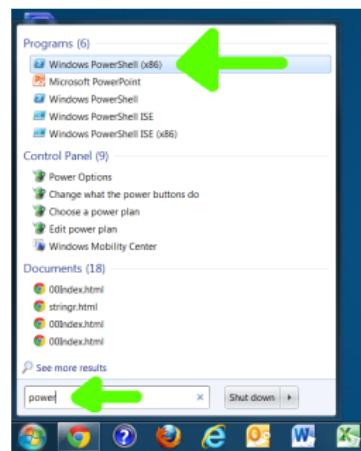
- all the files
- the entire history of snapshots of files states created each time you committed to the repository

Being a distributed version control system Git doesn't require you to be connected to the GitHub servers to work on your files and commit them to your branch of the repository.

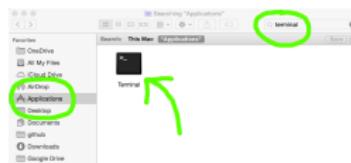
# Git on the Command Line

Accessing a command line interface in your OS of choice

MS Windows  
Open a 'PowerShell'



Mac OS X & later  
Open a 'Terminal'

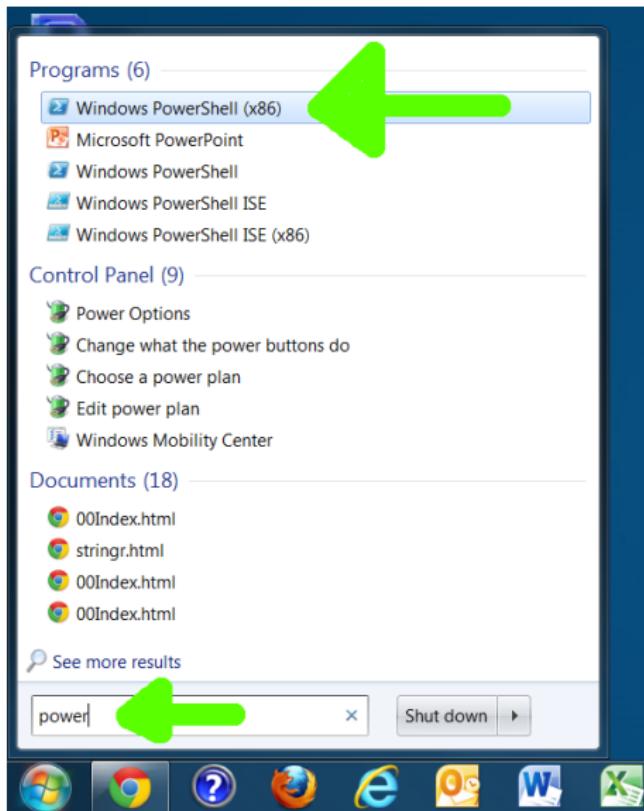


GNU+Linux  
Open a 'Terminal'



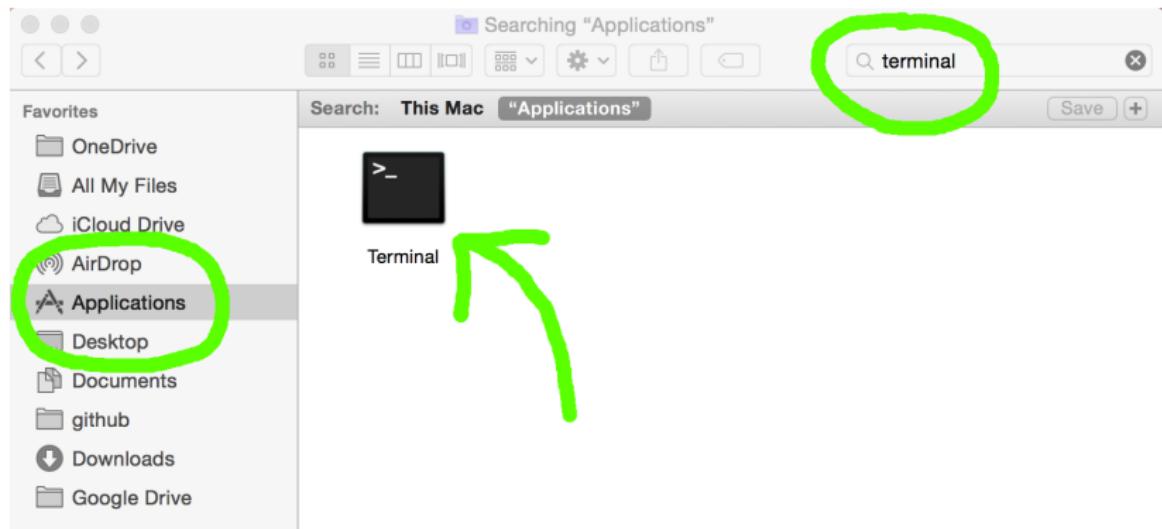
# Accessing a command line interface from MS Windows

Open a 'PowerShell' Powershell



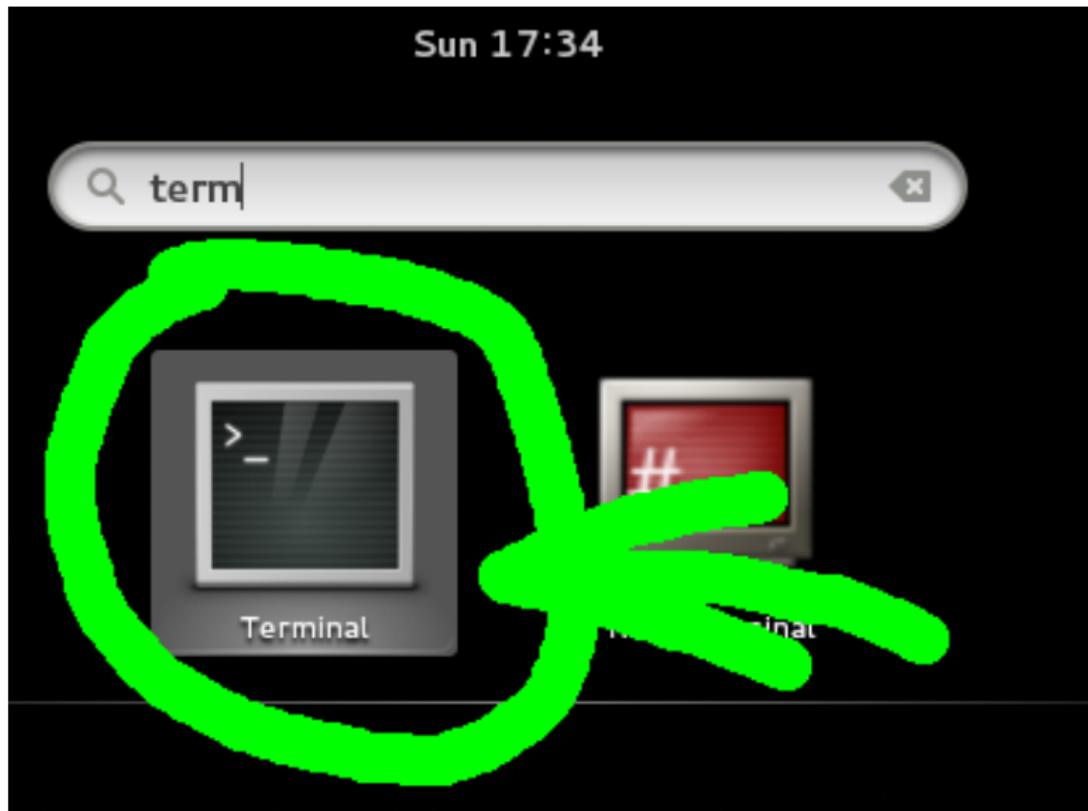
# Accessing a command line interface from MacOS

Open a 'Terminal'



# Accessing a command line interface from GNU+Linux

Open a 'Terminal'



# Configuring Git for the first time

## All Users

```
> git config --global user.name "Your Name"  
> git config --global user.email you@mail.com
```

# Configuring Git for the first time

## Choosing a Global Editor

The global editor is used to write your commit messages.

Default is Vim but Vim is a little heavy on keyboard short cuts for some...

Other Options:

- Notepad (available on all Windows PCs)
- Atom: <https://atom.io/docs/v1.0.0/getting-started-installing-atom>
- Sublime: <http://www.sublimetext.com/2>
- TextMate: <http://macromates.com/download>
- Emacs:  
`sudo apt-get install emacs`

# Configuring Git for the first time

<https://help.github.com/articles/associating-text-editors-with-git/>

Set a Text Editor of your choice (you'll need it installed)

Use one of the below

```
# Notepad:  
> git config --global core.editor notepad.exe  
# Atom:  
> git config --global core.editor "atom --wait"  
# Sublime:  
> git config --global core.editor "subl -n -w"  
# TextMate:  
> git config --global core.editor "mate -w"  
# Emacs:  
> git config --global core.editor "emacs"
```

# Cloning Your Repository from the GitHub Servers

Set the working directory to the location on your hard drive to which you would like the repository cloned:

## Powershell Users

```
> C:  
> cd C:\Users\username\Documents\GitHub_Repos\  
> dir
```

## Terminal Users

```
> cd /home/ben/Documents/GitHub_Repos  
> ls
```

# Copy HTTPS URL to use when cloning repository

The screenshot shows a GitHub repository page for 'brfitzpatrick / my\_new\_repo'. The page includes fields for 'Description' and 'Website', and displays metrics like 1 commit, 1 branch, 0 releases, and 1 contributor. A green arrow points from the bottom right towards the 'HTTPS clone URL' field, which contains the URL [https://github.com/brfitzpatrick/my\\_new\\_repo](https://github.com/brfitzpatrick/my_new_repo). Other options in this section include 'Copy to clipboard' and 'Download ZIP'.

This repository Search Pull requests Issues Gist

Unwatch 1 Star 0 Fork 0

brfitzpatrick / my\_new\_repo

Description Website

Short description of this repository Website for this repository (optional) Save or Cancel

1 commit 1 branch 0 releases 1 contributor

branch: master my\_new\_repo +

Initial commit

brfitzpatrick authored 33 minutes ago latest commit 4930beee35

LICENSE Initial commit 33 minutes ago

README.md Initial commit 33 minutes ago

README.md

my\_new\_repo

Code Issues 0

Pull requests 0

Viki

Pulse

Graphs

Settings

HTTPS clone URL  
https://github.com/brfitzpatrick/my\_new\_repo Copy to clipboard

You can clone with Subversion

Download ZIP

# Clone your Repository: GitHub Servers → your Hard Drive

Use the HTTPS URL you copied earlier:



## All Users

```
> git clone https://github.com/.../my_new_repo  
 .git  
> cd ./my_new_repo/  
> ls  
> git status
```

# Git

## The Three States of Files in a Git Tracked Directory

Files in a directory which you are version controlling with Git can exist in one of three states:

- **committed** data stored in the database for the associated repository
- **modified** local copy of a file (in your working directory) is different to the most recent version of that file stored in the database for the associated repository i.e. you have changed something but not committed the changes (yet)
- **staged** the local copy of a file which you have modified is marked to be added to the database in the next commit snapshot you send to the database

# Git

## Basic Git Workflow

- you add files or modify existing files in your local copy of the repository
- you stage these modified files ready to be committed to the database for the repository
- you perform a commit which takes the files as they were when staged and stores a snapshot of them in the database for the repository

# Git

## Adding New Files

- ① Open your favourite program for authoring R code (e.g. RStudio)
- ② Create an new R script
- ③ Save your R Script in the 'clone' of your repository

# Git

## Adding New Files

```
> git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what
  will be committed)
```

We have to tell Git that we want it to track the changes we make to this new file

# Git

## Adding New Files

We have to tell Git that we want it to track the changes we make to this new file

```
> git add filename.R
> 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:   filename.R
```

We can still change our mind about adding the file to the repository at this stage quite easily.

# Git

## Adding New Files

Once you're ready to 'commit' this change to the permanent record of changes in this repository we make a 'commit'

```
> git commit  
[master 311acfb] Initial Commit of some R code.  
1 files changed, 0 insertions(+), 0 deletions  
(-) create mode 100644 filename.R
```

You will be prompted to write a short description of the changes you are committing - these become invaluable as your project grows and if you are collaborating. Once you've written a commit message, save it (and close the text editor if you used one).

You should see confirmation of the commit immediately below the git commit line.

# Git

Now that you've committed some changes to your local clone of the repository this local version will be ahead of the version on the GitHub Server

```
> git status
On branch master
Your branch is ahead of 'origin/master' by 1
commit.
(use "git push" to publish your local commits)
```

Seeing as we are all currently connected to the internet, it's a good time to 'push' these changes to GitHub server ('origin/master' in the default setup) to bring the remote copy of this repository up to date.

## pushing to the remote servers

```
> git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 353 bytes | 0 bytes/s
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:brfitzpatrick/my_new_repo.git
 4930bee..99d61f1  master -> master
```

If you return to the page for your repository on the GitHub website and refresh the page you should see your most recent commit listed on the repository page.

## the whole process

Furthermore, now that we have ‘committed’ and ‘pushed’ our most recent changes our local copy of the repository and the remote copy of the repository on the GitHub servers will be identical

```
> git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

# Undoing Things Safely

## Checking out Previous Commits to a New Branch

To revert to a previous version of a file that you have 'snapshot' of from a previous 'commit'

Click on the file name on the GitHub website and click the 'History' button:

Provided we have committed all changes to the file then it is safe to use the checkout command which will overwrite the file in the current local working directory with the file from the previous 'commit'.

You can think of commits a bit like save points, if we have two we can move between them but if we load an old save without first making a current save we will lose the unsaved changes

# Git

## Undoing Things Safely

choose a commit you'd like to go back to...copy the SHA

```
# List existing branches of current directory
git branch

# checkout a previous commit (SHA = 58...a44)
> git checkout -b testing 589172506c13572be6cc05270
Switched to a new branch 'testing'

> git branch
  master
* testing
```

Now if you open your .R file you'll see that you have a previous version of the file without the new changes

# Git

## Undoing Things Safely

```
> git branch
# make some changes on testing branch
# push those changes upstream to server
> git push --set-upstream origin testing
# check website – your repo should now have two b
# Now to merge your changes from 'testing' back in
# to merge into a branch we must be on that branch
> git checkout master
# merge testing1 into current branch (master)
> git merge testing1
Auto-merging Car_Bar_Charts.R
CONFLICT (content): Merge conflict in Car_Bar_Ch
Automatic merge failed; fix conflicts and then co
```

# Resolving a Merge Conflict with a Merge Tool

```
git mergetool
```

This message is displayed because 'merge.tool' is n  
See 'git mergetool --tool-help' or 'git help config'  
'git mergetool' will now attempt to use one of the  
meld opendiff kdiff3 tkdiff xxdiff tortoisemerge gv  
Merging:

Car\_Bar\_Charts.R

Normal merge conflict for 'Car\_Bar\_Charts.R':

{local}: modified file

{remote}: modified file

Hit return to start merge resolution tool (meld):

# Resolving a Merge Conflict with a Merge Tool

Insert Screen shot of **Meld** in action

Now open you're R file to see the results of your merge

and open 'filename.R' and our changes are gone

checkout the most recent commit to get them back

if you want to go back to a previous version of a file and try something new without overwriting your most recent versions you can 'checkout' and old 'commit' to a new 'branch' of the repository

# Git

## Undoing things

One way to safely experiment with past versions of a file is to create a new ‘branch’ in the repository in which to do so

branches are separate lines of development of the same files

let’s make a new ‘testing3’ branch checking out a previous version of the file filename.R

```
> git checkout 99d61f19d85ae7aec691feb81ca8aef59f6a0
```

```
> git push --set-upstream origin testing3
```

we can then move between branches with the checkout command

```
> git checkout master
```

```
> git checkout testing3
```

# Merging Branches

To merge the 'master' and 'testing3' branches

```
> git checkout master  
> git merge testing3
```

if you get a merge conflict use a mergetool to resolve the conflict  
(you should have one installed by default from when you installed  
Git)

```
> git mergetool
```

# Forking

Fork a repository  
Use the Sync button in Windows/MacOS client  
to get new changes from server (and submit any changes you have made)

# Create a Repository, Add Some Files

- Log into your Github account
- Click the Repositories Tab
- Click the 'New' button (it's green)
- name your repository
- make it public (we can do a private one later)
- initialize repository with a README
- add a license GPL or MIT are easy FOSS licenses
- clone the repository to your hard drive
- add a file to your local clone of the repository
- commit it to the repository
- push your changes to the remote repository

## Fork a Friends Repository, Add Some Files

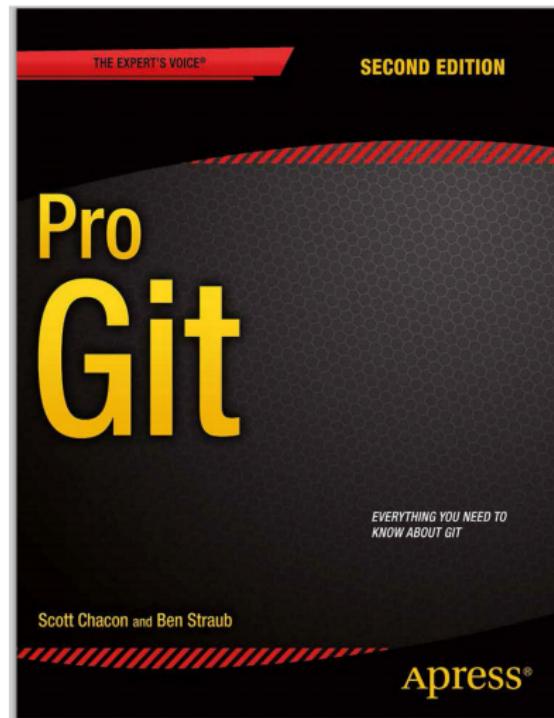
- search for a friend's github repository
- on their repository page click fork
- clone your fork of their repository to your hard drive
- edit one of their files
- push your changes to their repo

# Accepting and rejecting Pull Requests

Merging your friends' branches

# Recommended Further Reading

<http://git-scm.com/book/en/v2>

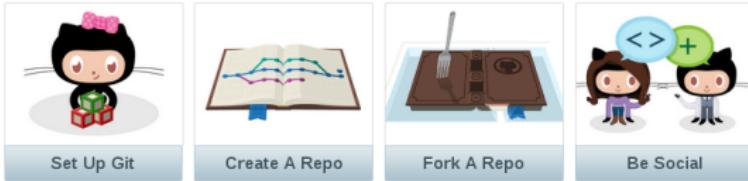


[?]

# Recommended Further Reading:

<https://help.github.com/>

**GitHub Help** Version ▾ Contact Support Return to GitHub



Set Up Git      Create A Repo      Fork A Repo      Be Social

Sometimes you just need a little **help**.

How can we help?

### Common Issues

- › Why are my contributions not showing up on my profile?
- › Why is Git always asking for my password?
- › Dealing with non-fast-forward errors
- › Error: Repository not found
- › Do you have custom plans?
- › HTTPS cloning errors
- › What is my disk quota?
- › What are the limits for viewing content and diffs in my repository?
- › Remove sensitive data
- › How do I access my organization account?

## References

# Image Credits

- Git Logo by Jason Long  
<http://git-scm.com/downloads/logos>
- GitHub Logo
- R Logo