Git

Jonathan Hodgson (Archie)

June 17, 2020

Aims

I am obviously not going to be able to go over everything that git does.

- I don’t know everything Git does
- Git does LOADS of stuff

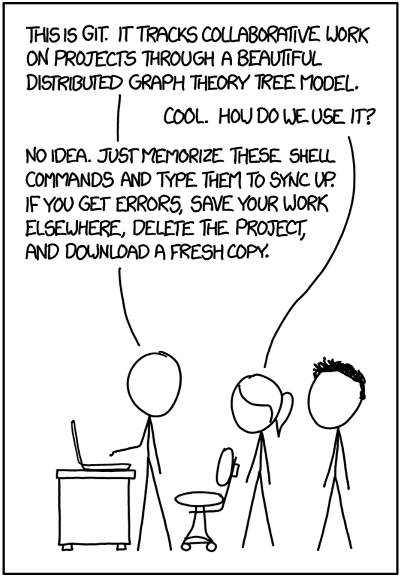
Hopefully after this you will be able to use Git well for most day-to-day tasks. Git has very compressive documentation. I hope that this will also give you enough of a background to understand the documentation.

What is Git

A very versatile Version Control System

- Keep track of source code (or other folders and files) and its history
- Facilitate collaboration
- Distributed

Obligitary XKCD Comic


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.

Install

```
# Ubuntu / Debian / Kali
sudo apt install git

# Centos / Fedora / Red Hat
sudo dnf install git

# Arch / Antergos / Manjaro
sudo pacman -S git

# Mac
brew install git

# Get the Version
git --version
```

Git for Windows: https://gitforwindows.org/

Setting It Up

User

```
git config --global user.name "Jonathan Hodgson"
git config --global user.email "git@jonathanh.co.uk"
```

Setting It Up

Editor

Pick One

```
# Set editor to vim
git config --global core.editor "vim"

# Set editor to nano
git config --global core.editor "nano"

# Set editor to VS Code
git config --global core.editor "code -w"

# Set editor to Sublime
git config --global core.editor "subl -w"
```

Terminology

Objects

Blob In Git, a file is called a blob.

Tree In Git, a directory is called a tree.

Commit A snapshot of your code

Naïve Approach

```
Project
 ├─ draft
 │   ├─ some
 │   └─ files
 ├─ final-draft
 │   ├─ some
 │   └─ files
 └─ final
     ├─ some
     └─ files
```

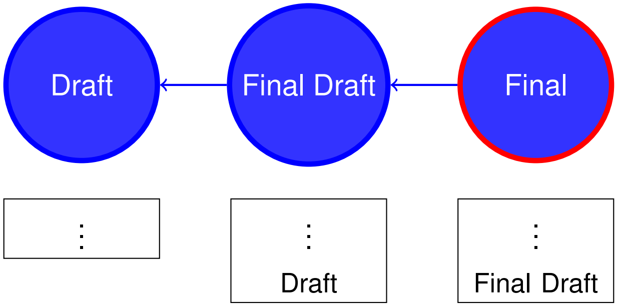
Pros

- Simple
- No dependencies
- No Learning curve

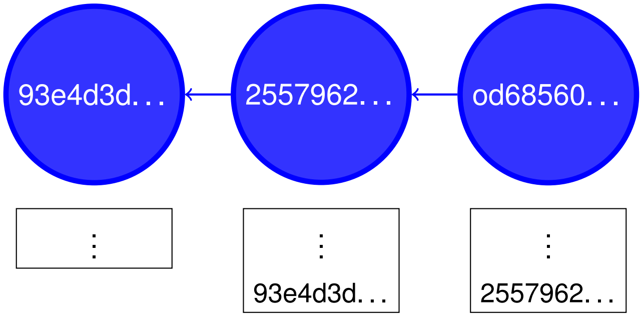
Cons

- Difficult to collaborate
- Lot’s of wasted disk space
- Can be difficult to work out chronological order

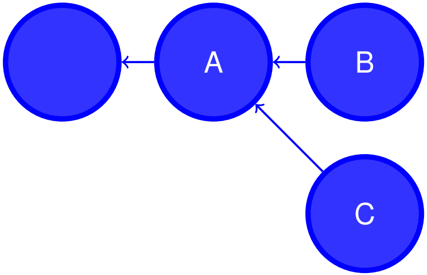
Model it



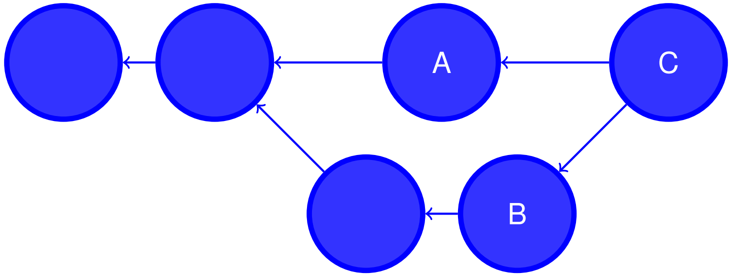
Commits



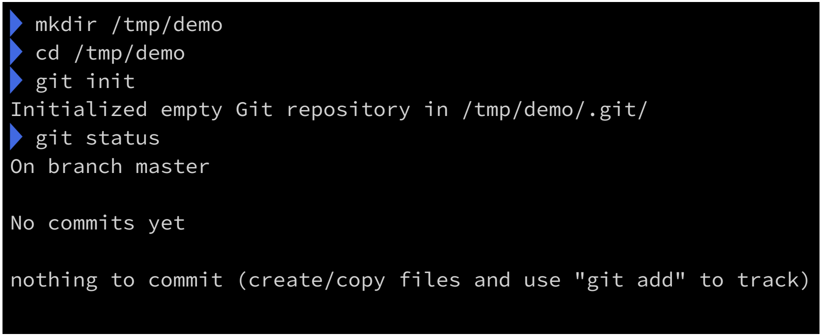
Commits / Branches



Commits / Branches

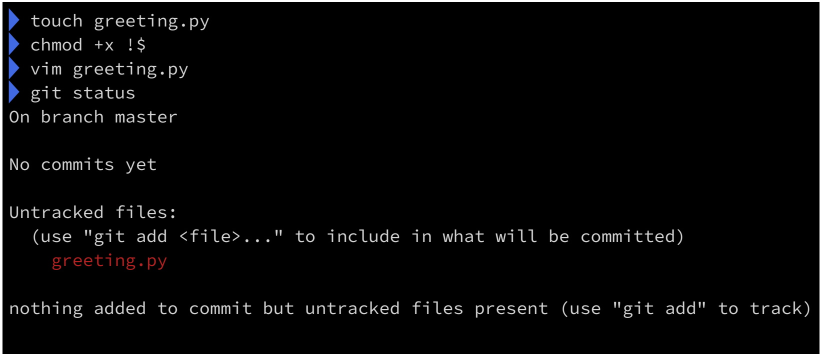


Create a repository



```
▶ mkdir /tmp/demo
▶ cd /tmp/demo
▶ git init
Initialized empty Git repository in /tmp/demo/.git/
▶ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Git status

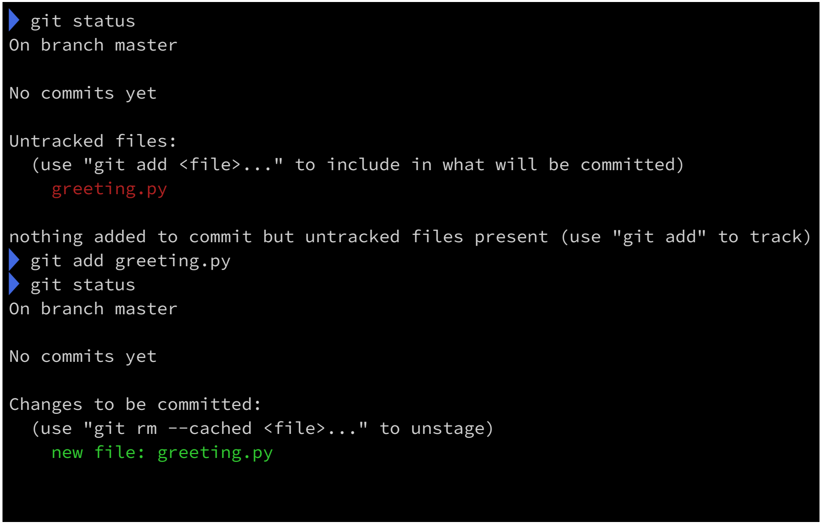


```
▶ touch greeting.py
▶ chmod +x !$
▶ vim greeting.py
▶ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
```

Staging Area

```
# Add files / or directories
git add <file|directory> [<file|directory>...]
# Add everything not in gitignore
git add -A
```

Staging Area



```
▶ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
▶ git add greeting.py
▶ git status
On branch master

No commits yet

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

Committing

```
git commit
```

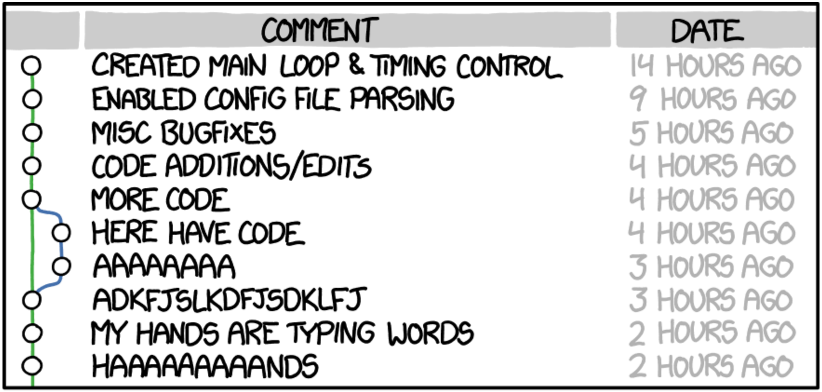
- First line should be concise summary around 50 chars
- Body Should be wrapped to around 70 chars
- There should be an empty line separating summary from body
- If contributing to a project, check per-project guidelines
  - Normally in contributing.md or similar
- Use the imperative: “Fix bug” and not “Fixed bug” or “Fixes bug.”

When should you commit?

Commit early, commit often

- Every time you complete a small change or fix a bug
- You don't normally want to commit broken code (intentionally at least)
- In some instances you might want to auto-commit - but probably not too often.
  - Normally this works if changes can't break something. E.g. Password Manager

Commit Messages

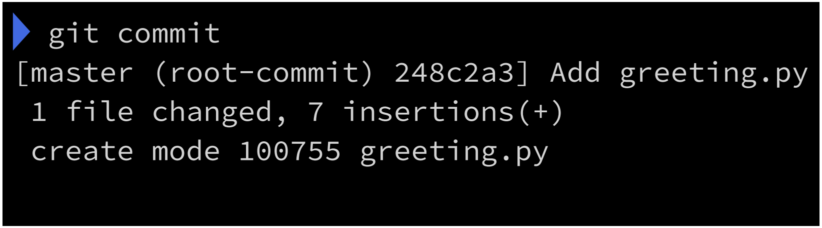


	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS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

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

Commit

```
# Open editor for message
git commit
# Read message from file
git commit -F <file or - for stdin>
# Provide message directly
git commit -m "<message>"
```

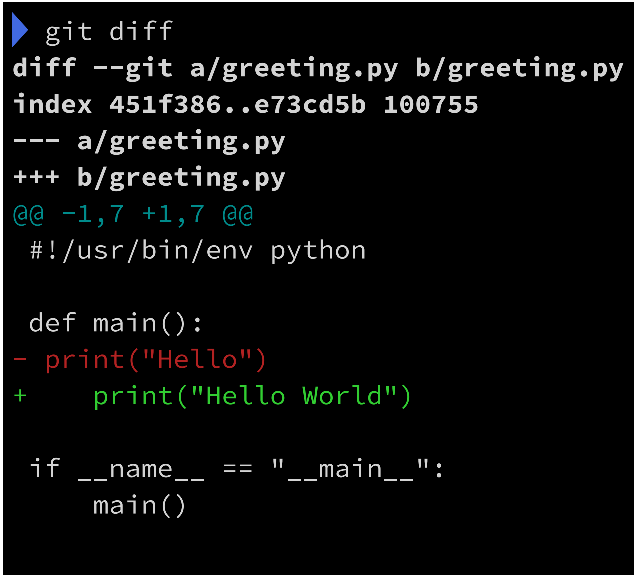


```
▶ git commit
[master (root-commit) 248c2a3] Add greeting.py
 1 file changed, 7 insertions(+)
 create mode 100755 greeting.py
```

Diff

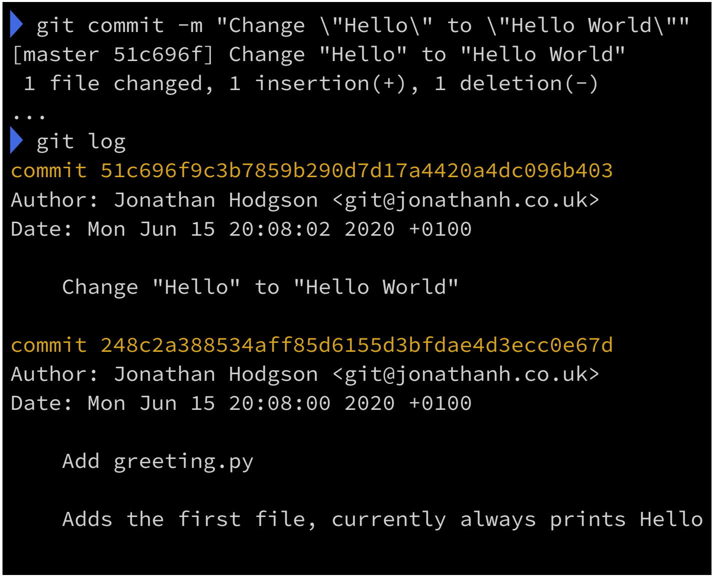
```
# Diff between last commit and current state
git diff
# Diff between 2 commits or references
git diff commit1..commit2
# Same as above but on a single file
git diff a/file
```

Diff



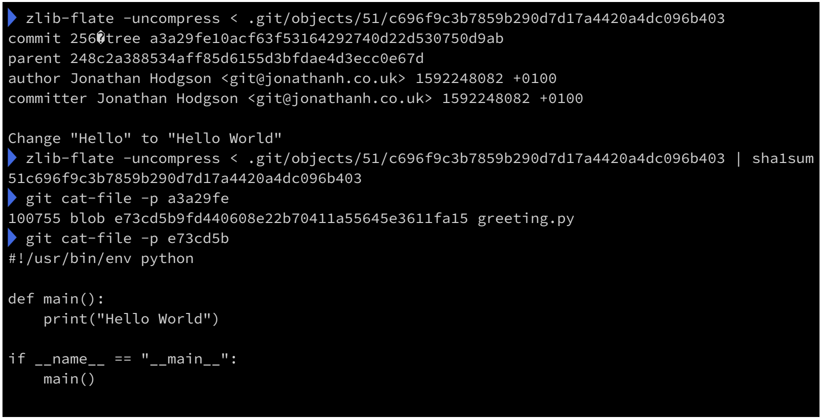
```
 git diff
diff --git a/greeting.py b/greeting.py
index 451f386..e73cd5b 100755
--- a/greeting.py
+++ b/greeting.py
@@ -1,7 +1,7 @@
 #!/usr/bin/env python

 def main():
- print("Hello")
+    print("Hello World")

 if __name__ == "__main__":
     main()
```

Log



```
❯ git commit -m "Change \"Hello\" to \"Hello World\""
[master 51c696f] Change "Hello" to "Hello World"
 1 file changed, 1 insertion(+), 1 deletion(-)
...
❯ git log
commit 51c696f9c3b7859b290d7d17a4420a4dc096b403
Author: Jonathan Hodgson <git@jonathanh.co.uk>
Date: Mon Jun 15 20:08:02 2020 +0100

    Change "Hello" to "Hello World"

commit 248c2a388534aff85d6155d3bfdae4d3ecc0e67d
Author: Jonathan Hodgson <git@jonathanh.co.uk>
Date: Mon Jun 15 20:08:00 2020 +0100

    Add greeting.py

    Adds the first file, currently always prints Hello
```

Under the hood



```
▶ zlib-flate -uncompress < .git/objects/51/c696f9c3b7859b290d7d17a4420a4dc096b403
commit 256�tree a3a29fe10acf63f53164292740d22d530750d9ab
parent 248c2a388534aff85d6155d3bfdae4d3ecc0e67d
author Jonathan Hodgson <git@jonathanh.co.uk> 1592248082 +0100
committer Jonathan Hodgson <git@jonathanh.co.uk> 1592248082 +0100

Change "Hello" to "Hello World"
▶ zlib-flate -uncompress < .git/objects/51/c696f9c3b7859b290d7d17a4420a4dc096b403 | sha1sum
51c696f9c3b7859b290d7d17a4420a4dc096b403
▶ git cat-file -p a3a29fe
100755 blob e73cd5b9fd440608e22b70411a55645e3611fa15 greeting.py
▶ git cat-file -p e73cd5b
#!/usr/bin/env python

def main():
    print("Hello World")

if __name__ == "__main__":
    main()
```

.gitignore

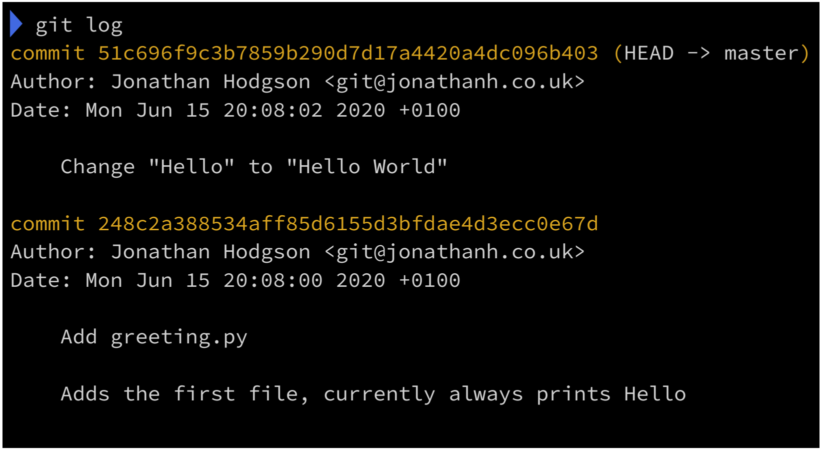
This file tells git which files not to track.

```
*.log
*.doc
*.pem
*.docx
*.jpg
*.jpeg
*.pdf
*.png
.DS_Store/
*.min.css
*.min.js
dist/
```

References

- We have just seen that commits are simply (compressed) text files, addressed by a hash.
- References are a way of addressing them without remembering the hash.
- Unlike the hashes, references can change - and they do change.

References



```
▶ git log
commit 51c696f9c3b7859b290d7d17a4420a4dc096b403 (HEAD -> master)
Author: Jonathan Hodgson <git@jonathanh.co.uk>
Date: Mon Jun 15 20:08:02 2020 +0100

    Change "Hello" to "Hello World"

commit 248c2a388534aff85d6155d3bfdae4d3ecc0e67d
Author: Jonathan Hodgson <git@jonathanh.co.uk>
Date: Mon Jun 15 20:08:00 2020 +0100

    Add greeting.py

    Adds the first file, currently always prints Hello
```

References

```
▶ cat .git/refs/heads/master
51c696f9c3b7859b290d7d17a4420a4dc096b403
▶ cat .git/HEAD
ref: refs/heads/master
```

- References are stored in the `.git/refs` folder
- The `heads` folder contains references to the heads (or tips) of all local branches

References
HEAD

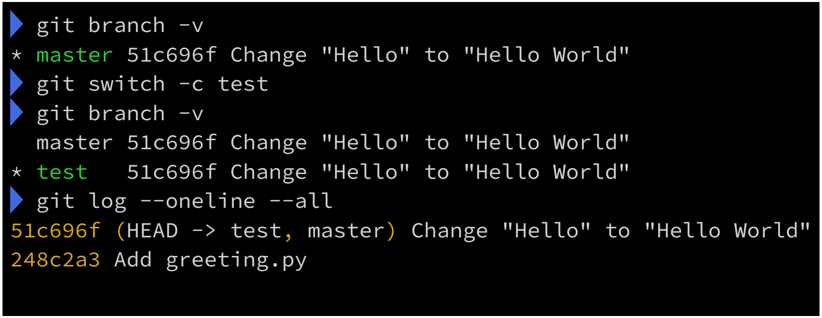
- The HEAD references is directly in the `.git` folder.
- It refers to the “current” commit. It is how git knows where you are.
- This normally refers to a branch’s head commit.
- In some situations it will refer to a commit directly.

Branches

- Allows multiple features to be developed in parallel without interference.
- Allows multiple people to collaborate easily.

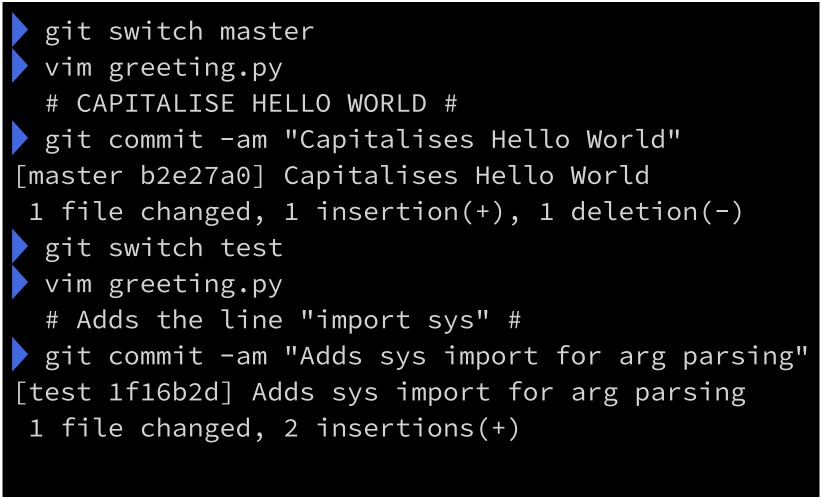
```
# List Branches
git branch # -v adds more info
# Create a branch called test
git branch test # or
cp ~/.git/refs/heads/master ~/.git/refs/heads/test
# Switch to new branch
git switch test # or
git checkout test
# Create and switch in one go
git switch -c test # or
git checkout -b test
```

Branches



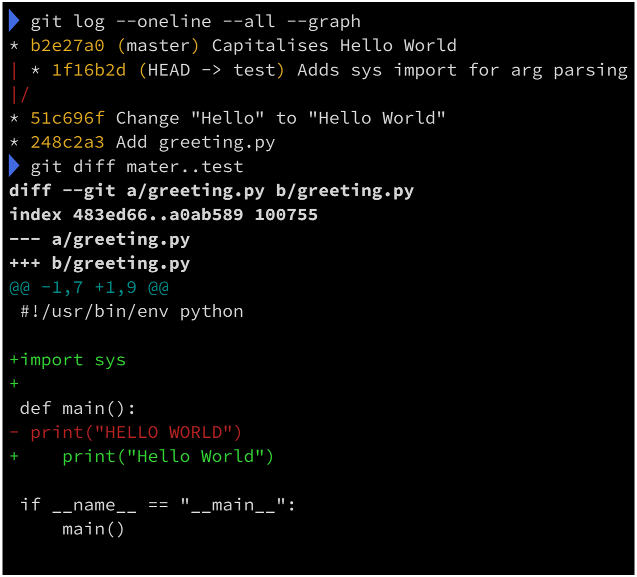
```
▶ git branch -v
* master 51c696f Change "Hello" to "Hello World"
▶ git switch -c test
▶ git branch -v
  master 51c696f Change "Hello" to "Hello World"
* test   51c696f Change "Hello" to "Hello World"
▶ git log --oneline --all
51c696f (HEAD -> test, master) Change "Hello" to "Hello World"
248c2a3 Add greeting.py
```

Differing Branches



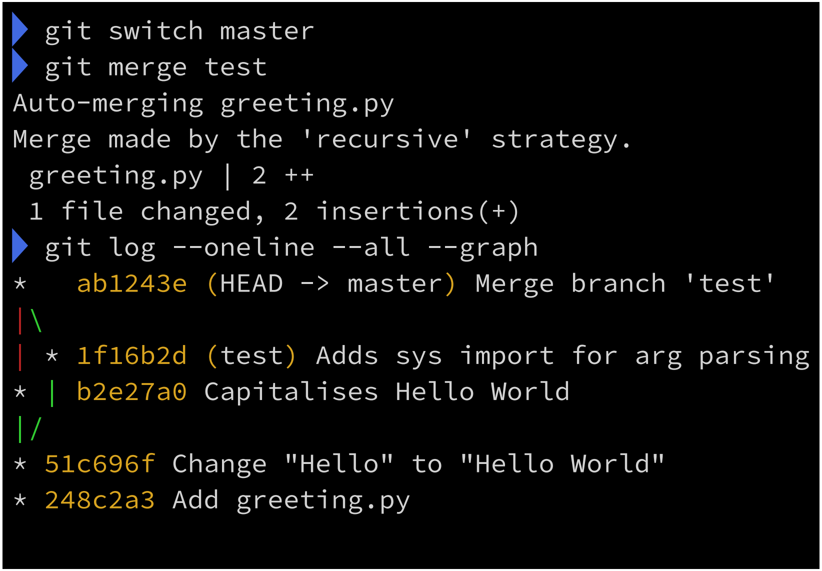
```
▶ git switch master
▶ vim greeting.py
  # CAPITALISE HELLO WORLD #
▶ git commit -am "Capitalises Hello World"
[master b2e27a0] Capitalises Hello World
 1 file changed, 1 insertion(+), 1 deletion(-)
▶ git switch test
▶ vim greeting.py
  # Adds the line "import sys" #
▶ git commit -am "Adds sys import for arg parsing"
[test 1f16b2d] Adds sys import for arg parsing
 1 file changed, 2 insertions(+)
```

Differing Branches



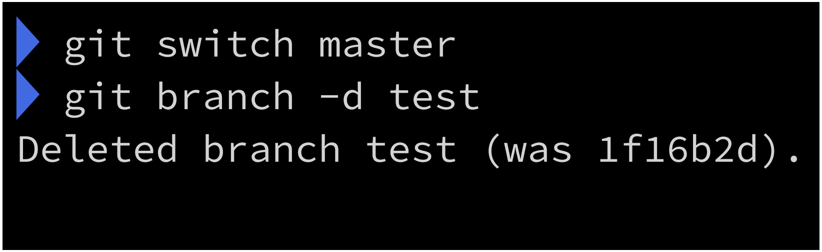
```
▶ git log --oneline --all --graph
* b2e27a0 (master) Capitalises Hello World
| * 1f16b2d (HEAD -> test) Adds sys import for arg parsing
|/
* 51c696f Change "Hello" to "Hello World"
* 248c2a3 Add greeting.py
▶ git diff mater..test
diff --git a/greeting.py b/greeting.py
index 483ed66..a0ab589 100755
--- a/greeting.py
+++ b/greeting.py
@@ -1,7 +1,9 @@
 #!/usr/bin/env python

+import sys
+
 def main():
- print("HELLO WORLD")
+    print("Hello World")

 if __name__ == "__main__":
     main()
```

Simple Merge



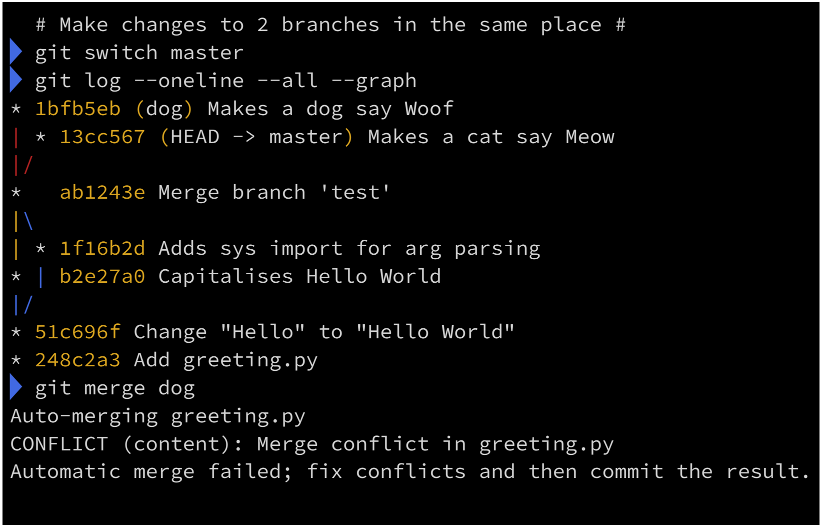
```
▶ git switch master
▶ git merge test
Auto-merging greeting.py
Merge made by the 'recursive' strategy.
 greeting.py | 2 ++
 1 file changed, 2 insertions(+)
▶ git log --oneline --all --graph
*   ab1243e (HEAD -> master) Merge branch 'test'
|\
| * 1f16b2d (test) Adds sys import for arg parsing
* | b2e27a0 Capitalises Hello World
|/
* 51c696f Change "Hello" to "Hello World"
* 248c2a3 Add greeting.py
```

Tidy Up



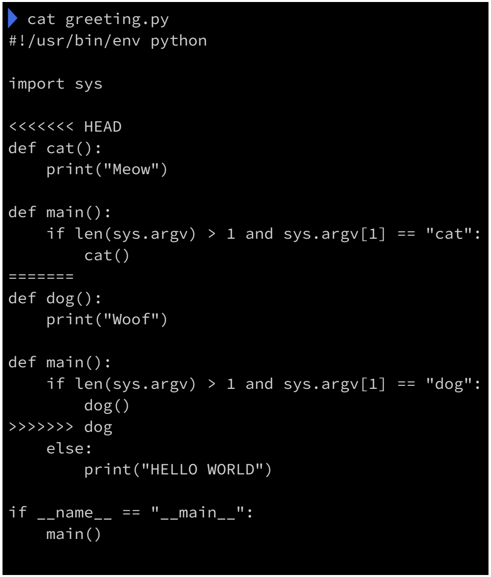
```
▶ git switch master
▶ git branch -d test
Deleted branch test (was 1f16b2d).
```

More Complex merge



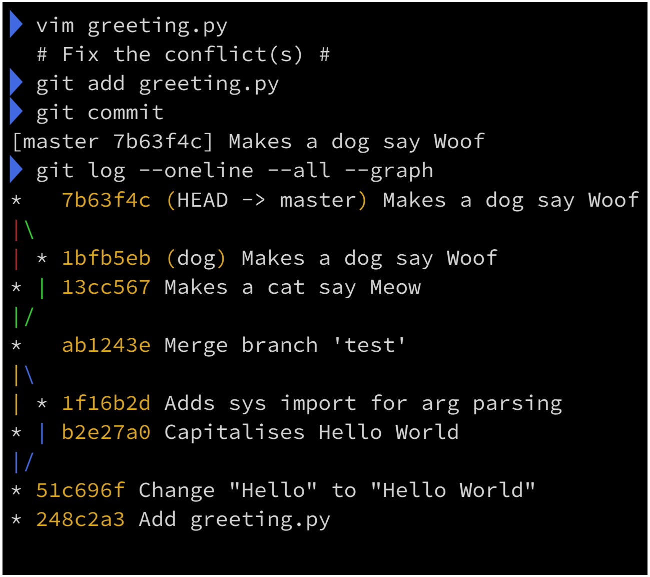
```
  # Make changes to 2 branches in the same place #
▶ git switch master
▶ git log --oneline --all --graph
* 1bfb5eb (dog) Makes a dog say Woof
| * 13cc567 (HEAD -> master) Makes a cat say Meow
|/
*   ab1243e Merge branch 'test'
|\
| * 1f16b2d Adds sys import for arg parsing
* | b2e27a0 Capitalises Hello World
|/
* 51c696f Change "Hello" to "Hello World"
* 248c2a3 Add greeting.py
▶ git merge dog
Auto-merging greeting.py
CONFLICT (content): Merge conflict in greeting.py
Automatic merge failed; fix conflicts and then commit the result.
```

More Complex merge



```
▶ cat greeting.py
#!/usr/bin/env python

import sys

<<<<<<< HEAD
def cat():
    print("Meow")

def main():
    if len(sys.argv) > 1 and sys.argv[1] == "cat":
        cat()
=======
def dog():
    print("Woof")

def main():
    if len(sys.argv) > 1 and sys.argv[1] == "dog":
        dog()
>>>>>>> dog
    else:
        print("HELLO WORLD")

if __name__ == "__main__":
    main()
```

More Complex merge



```
▶ vim greeting.py
  # Fix the conflict(s) #
▶ git add greeting.py
▶ git commit
[master 7b63f4c] Makes a dog say Woof
▶ git log --oneline --all --graph
*   7b63f4c (HEAD -> master) Makes a dog say Woof
|\
| * 1bfb5eb (dog) Makes a dog say Woof
* | 13cc567 Makes a cat say Meow
|/
*   ab1243e Merge branch 'test'
|\
| * 1f16b2d Adds sys import for arg parsing
* | b2e27a0 Capitalises Hello World
|/
* 51c696f Change "Hello" to "Hello World"
* 248c2a3 Add greeting.py
```

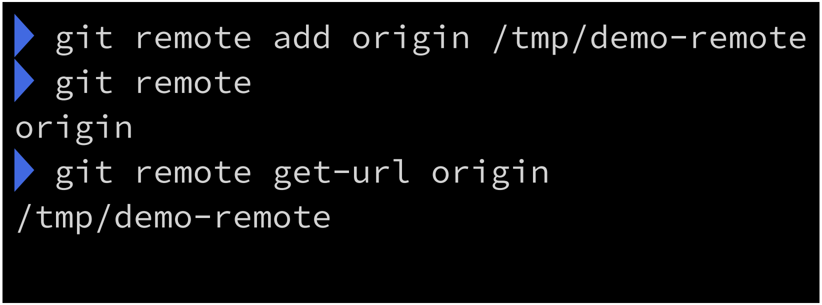
Time Travel

```
# Print a version of a file
git show <commit or reference>:<file>
# Restore a file from a previous version
git restore -s <commit or reference> file # or
git checkout <commit or reference> -- file
# Go back in time to a commit
git switch --detach <commit or reference> # or
git checkout <commit or reference>
```

Remotes

- The majority of Git commands only affect your local repository.
- Git has a concept called remotes which you can think of as other instances of the same repository
- Git has a selection of commands that are used to communicate with these remote repositories
- It can communicate on multiple protocols including
  - HTTP(S)
  - SSH
  - GIT
  - Local Filesystem

Adding a remote



```
▶ git remote add origin /tmp/demo-remote
▶ git remote
origin
▶ git remote get-url origin
/tmp/demo-remote
```

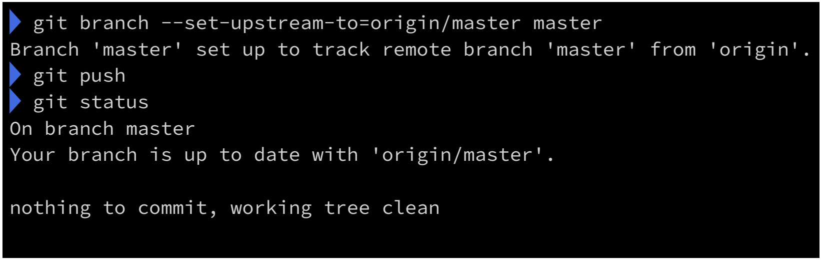
Pushing your code

Long Way

```
git push <remote> <local-branch>:<remote-branch>
# E.g.
git push origin master:master
```

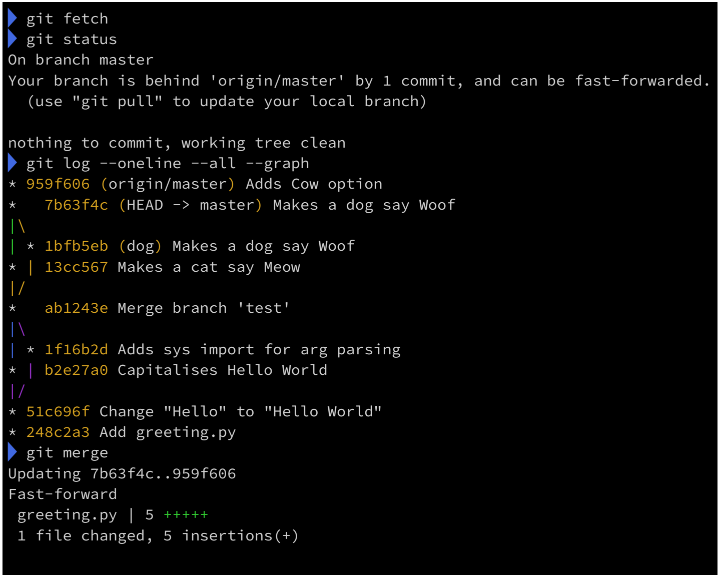
Pushing your code

Easy way



```
▶ git branch --set-upstream-to=origin/master master
Branch 'master' set up to track remote branch 'master' from 'origin'.
▶ git push
▶ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Retrieving changes from the remote



```
❯ git fetch
❯ git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
❯ git log --oneline --all --graph
* 959f606 (origin/master) Adds Cow option
*   7b63f4c (HEAD -> master) Makes a dog say Woof
|\
| * 1bfb5eb (dog) Makes a dog say Woof
* | 13cc567 Makes a cat say Meow
|/
*   ab1243e Merge branch 'test'
|\
| * 1f16b2d Adds sys import for arg parsing
* | b2e27a0 Capitalises Hello World
|/
* 51c696f Change "Hello" to "Hello World"
* 248c2a3 Add greeting.py
❯ git merge
Updating 7b63f4c..959f606
Fast-forward
 greeting.py | 5 +++++
 1 file changed, 5 insertions(+)
```

Git Pull

Shortcut

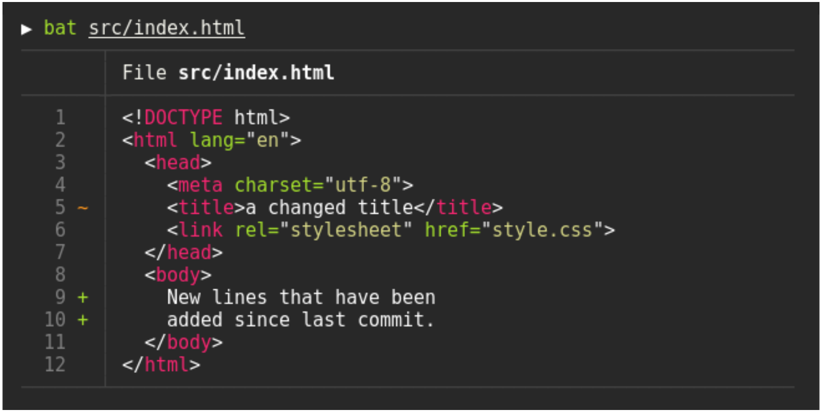
```
git pull
git pull <remote> <branch>
```

Cloning

```
# Clone a repository into a folder
git clone <URL> <folder>

# Clone a repository into a folder on a specific branch
git clone --branch <branch> <URL> <folder>

# Shallow clone a repository into a folder
git clone --shallow <URL> <folder>
```

Useful supporting tools

Bat



```
▶ bat src/index.html
───────┬──────────────────────────────────────────
       │ File src/index.html
───────┼──────────────────────────────────────────
       │ <!DOCTYPE html>
       │ <html lang="en">
       │   <head>
       │     <meta charset="utf-8">
     ~ │     <title>a changed title</title>
       │     <link rel="stylesheet" href="style.css">
       │   </head>
       │   <body>
     + │     New lines that have been
     + │     added since last commit.
       │   </body>
       │ </html>
───────┴──────────────────────────────────────────
```

https://github.com/sharkdp/bat

Useful supporting tools

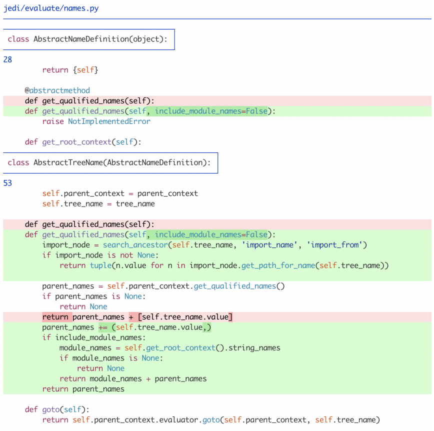
RigGrep / Fd

https://github.com/sharkdp/fd

https://github.com/BurntSushi/ripgrep

Useful supporting tools

Delta



```
jedi/evaluate/names.py

class AbstractNameDefinition(object):

28
        return {self}

    @abstractmethod
    def get_qualified_names(self):
    def get_qualified_names(self, include_module_names=False):
        raise NotImplementedError

    def get_root_context(self):

class AbstractTreeName(AbstractNameDefinition):

53
        self.parent_context = parent_context
        self.tree_name = tree_name

    def get_qualified_names(self):
    def get_qualified_names(self, include_module_names=False):
        import_node = search_ancestor(self.tree_name, 'import_name', 'import_from')
        if import_node is not None:
            return tuple(n.value for n in import_node.get_path_for_name(self.tree_name))

        parent_names = self.parent_context.get_qualified_names()
        if parent_names is None:
            return None
        return parent_names + [self.tree_name.value]
        parent_names += (self.tree_name.value,)
        if include_module_names:
            module_names = self.get_root_context().string_names
            if module_names is None:
                return None
            return module_names + parent_names
        return parent_names

    def goto(self):
        return self.parent_context.evaluator.goto(self.parent_context, self.tree_name)
```

https://github.com/dandavison/delta

Useful supporting tools

BFG Repo Cleaner

You'll need something like this when you realise you have just committed your ssh keys

https://rtyley.github.io/bfg-repo-cleaner/

Useful supporting tools

Shell Integration

Git ships with completion for bash, zsh and tcsh. You may need to source it in the relevant rc file.

Prompt customisation is available out of the box for bash and zsh.

Useful supporting tools

Pass

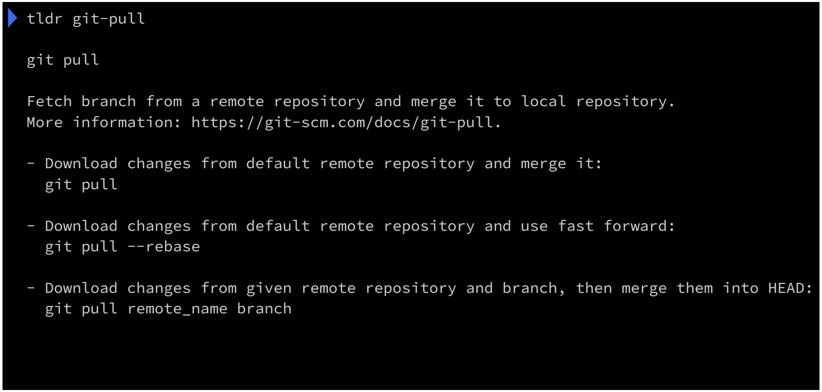
- Password Manager
- Uses Git for keeping track of history
- Syncs using Git
- Everything is encrypted with a GPG key
- Has compatible android, ios and browser apps.

https://www.passwordstore.org/

Useful supporting tools

tldr

The man page for git pull is over 700 lines.



```
▶ tldr git-pull

  git pull

  Fetch branch from a remote repository and merge it to local repository.
  More information: https://git-scm.com/docs/git-pull.

  - Download changes from default remote repository and merge it:
    git pull

  - Download changes from default remote repository and use fast forward:
    git pull --rebase

  - Download changes from given remote repository and branch, then merge them into HEAD:
    git pull remote_name branch
```

https://github.com/tldr-pages/tldr