

git good

Learn to use git for version control

Presented by Joel C
(slides adapted from Edmund G)
(tweaked 2025 by Raven)



University of Warwick
Computing Society



git

Has this ever happened to you?



Copy Of
EssayFINALFINAL
2.doc



Essay.doc



Essay2.doc



Essay2withconclu
sion.doc



Essay-FINAL.doc



EssayFINALFINAL
doc



EssayFINALFINAL
2.doc

Why is this bad?

- Multiple copies of nearly the same thing
 - Need to remember which one is the latest one
- Gets worse with multiple files
- Gets *even* worse with multiple people

So why do we do it?

- We might care about the history of a file
 - Especially important when working in a group
- We might want to experiment with changes
 - (and get back to the old state if it didn't work)

Is there a better way?

Version control

- Software that tracks changes made to files
 - Most often used for source code, but can be anything
- Lots of different ones
 - CVS, SVN, Bazaar, Perforce, Mercurial



git

Why git?

- It's really popular!
 - ~90% market share
 - Many open source projects and most (tech) companies use it
- It's ergonomic and has powerful features
 - Cheap local branching, distributed model, ...
 - Easier to use/learn than competitors
 - <https://z.github.io/whygitisbetter/>

About git

- Created by Linus Torvalds for use developing the Linux kernel
- It's free, and open source!
- Over 18 years old
 - First released in July 2005
 - Older than the majority of you



Aim of this talk

- Give an introduction to *basic* git
- Convince you that git is worth using

The command line

- Who's used the command line before?
- Git uses subcommands
 - `git <subcommand> <flags> <arguments>`
- Can easily find documentation with
 - `man git <subcommand>`

Repositories

- Repositories are just folders managed by git
 - Can be thought of as a project
 - Tracks changes over time
 - Also called a “repo”
- `git init` initialises a repository in the current folder
- Internal workings stored in the `.git` folder
 - *Don't touch this!*

```
[uwcs@hopper:~/git-good]$ ls -a
```

```
.  ..
```

```
[uwcs@hopper:~/git-good]$ git init
```

```
Initialized empty Git repository in /home/uwcs/git-good/.git/
```

```
[uwcs@hopper:~/git-good]$ ls -a
```

```
.  ..  .git
```

```
[uwcs@hopper:~/git-good]$ █
```

Dark mode users after
accidentally turning on light mode:



Blind Myself With a Lamp For No
Reason!!

33 views

Working directory

- The "state" of your project
 - What you see when you type `ls` (excluding `.git` folder)
 - If there are no changes since the last commit, we say it is "clean"
- We make changes in the working directory as we develop our code
- We can see what has changed in the working directory with the `git status` command

```
[uwcs@hopper:~/git-good]$ git status
```

```
On branch main
```

```
No commits yet
```

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

```
[uwcs@hopper:~/git-good]$ █
```


What is a commit?

- Commits are snapshots in history of the repository
- "Named" by hashes of their content
 - Commits can be referred to by that hash

```
1     a = 0
2     b = 1
3
4     output = f"{a}, {b}"
5
6     for i in range(3, 11):
7         a, b = b, a + b
8         output += f", {b}"
9
10    print(output)
```

- [uwcs@hopper:~/git-good]\$ ls -a
 . .. fibonacci.py .git

- [uwcs@hopper:~/git-good]\$ python3 fibonacci.py
0, 1, 1, 2, 3, 5, 8, 13, 21, 34

- [uwcs@hopper:~/git-good]\$ git status
On branch main

No commits yet

Untracked files:

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

fibonacci.py

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

- [uwcs@hopper:~/git-good]\$ █

Staging area

- The changes we want to include in the next commit
 - Sometimes we only want to pick some of the changes!
- We can add things to the staging area with the `git add` command
- These will also show up when we re-run `git status`

- [uwcs@hopper:~/git-good]\$ git add fibonacci.py

- [uwcs@hopper:~/git-good]\$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: fibonacci.py

- [uwcs@hopper:~/git-good]\$ ls -a

. .. fibonacci.py .git

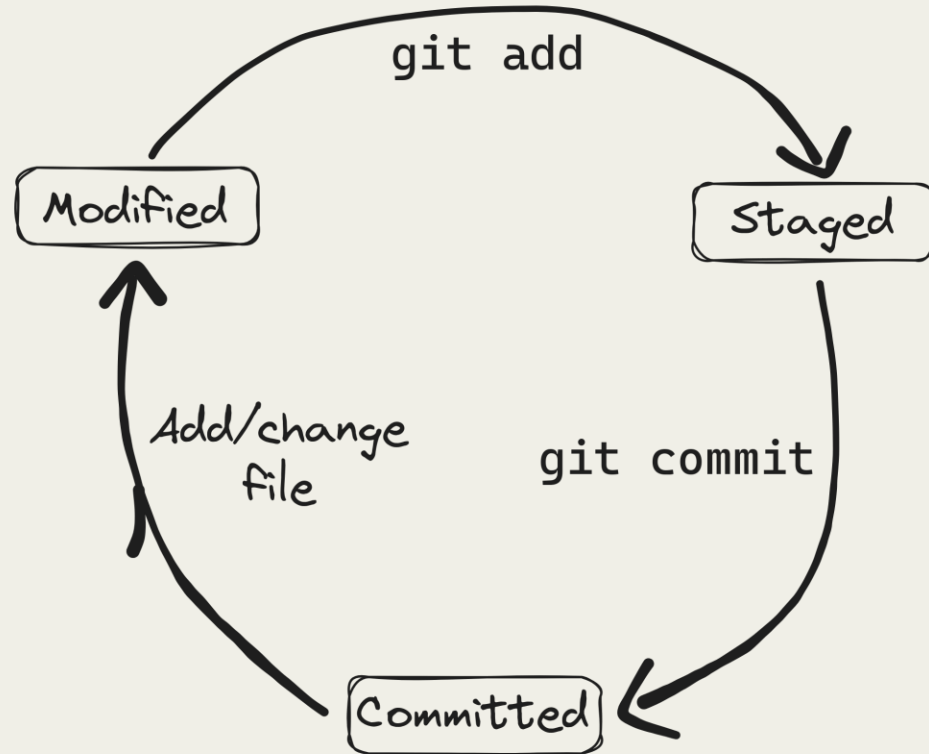
- [uwcs@hopper:~/git-good]\$

Making a commit

- Taking a snapshot of the changes we picked in our staging area
- Use the `git commit` command to make one
 - Use the `-m` flag to give them short messages
 - Should be an imperative phrase

- [uwcs@hopper:~/git-good]\$ git commit -m "Added fibonacci program"
[main (root-commit) 58aad9b] Added fibonacci program
1 file changed, 10 insertions(+)
create mode 100644 fibonacci.py
- [uwcs@hopper:~/git-good]\$ git status
On branch main
nothing to commit, working tree clean
- [uwcs@hopper:~/git-good]\$ ls -a
. .. fibonacci.py .git
- [uwcs@hopper:~/git-good]\$ █

The three stages of a file





fibonacci.py

```
1  a = 0
2  b = 1
3
4  output = f"{a}, {b}"
5
6  for i in range(3, 16):
7      a, b = b, a + b
8      output += f", {b}"
9
10 print(output)
```

more on this
later



① README.md >  # Git Good Demo

1 **# Git Good Demo**

2

3 This program prints out Fibonacci numbers

4

5 This was written for UWCS' Git Good talk

- `[uwcs@hopper:~/git-good]$ ls -a`
`. .. fibonacci.py .git README.md`
- `[uwcs@hopper:~/git-good]$ python3 fibonacci.py`
`0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377`
- `[uwcs@hopper:~/git-good]$ git status`
On branch main
Changes not staged for commit:
 (use "git add <file>..." to update what will be committed)
 (use "git restore <file>..." to discard changes in working directory)
 modified: fibonacci.py

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

no changes added to commit (use "git add" and/or "git commit -a")
- `[uwcs@hopper:~/git-good]$` █

- [uwcs@hopper:~/git-good]\$ git add fibonacci.py README.md
 - [uwcs@hopper:~/git-good]\$ git add --all
- } equivalent
- [uwcs@hopper:~/git-good]\$ git status
On branch main
Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 new file: README.md
 modified: fibonacci.py
 - [uwcs@hopper:~/git-good]\$ ls -a
 . .. fibonacci.py .git README.md
 - [uwcs@hopper:~/git-good]\$ █

- `[uwcs@hopper:~/git-good]$ git commit -m "Updated range and added README"`
`[main 9e5bfb3] Updated range and added README`
`2 files changed, 6 insertions(+), 1 deletion(-)`
`create mode 100644 README.md`
- `[uwcs@hopper:~/git-good]$ git status`
`On branch main`
`nothing to commit, working tree clean`
- `[uwcs@hopper:~/git-good]$` █

Recap so far

- Making repositories with `git init`
- Looking at their state with `git status`
- Adding to the staging area with `git add`
- Taking snapshots of history with `git commit`

Questions?

Looking at histories with log

- The `git log` command lets us look back on our commit history
- We can use some flags to make it look prettier
 - `--color --oneline --graph --decorate --all`
 - Will use these in all examples going forward

- [uwcs@hopper:~/git-good]\$ git log
commit 9e5bfb36913c29bf6e409dd3ff0cfd54f874f5cd (HEAD -> main)
Author: uwcs <exec@uwcs.co.uk>
Date: Tue Sep 10 17:38:55 2024 +0100

Updated range and added README

```
commit 58aad9b3289bd4bfa286f14b324129706dfd74ce
Author: uwcs <exec@uwcs.co.uk>
Date: Tue Sep 10 17:30:39 2024 +0100
```

Added fibonacci program

- [uwcs@hopper:~/git-good]\$ git log --color --oneline --graph --decorate --all
 - * 9e5bfb3 (HEAD -> main) Updated range and added README
 - * 58aad9b Added fibonacci program
- [uwcs@hopper:~/git-good]\$

Time travelling with checkout

- The `git checkout` command lets us travel around the history of our repo
- `git checkout <name>` lets us visit commits
 - Working directory must be clean, otherwise we'd lose our changes!
- HEAD is a synonym for the current location history
 - Don't need to remember long hashes
- HEAD~n means the nth previous commit

- [uwcs@hopper:~/git-good]\$ ls -a
. .. fibonacci.py .git README.md

- [uwcs@hopper:~/git-good]\$ git checkout HEAD

- [uwcs@hopper:~/git-good]\$ ls -a
. .. fibonacci.py .git README.md

- [uwcs@hopper:~/git-good]\$ git checkout HEAD~1
Note: switching to 'HEAD~1'.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 58aad9b Added fibonacci program

- [uwcs@hopper:~/git-good]\$ ls -a
. .. fibonacci.py .git

• [uwcs@hopper:~/git-good]\$ cat fibonacci.py

```
a = 0
```

```
b = 1
```

```
output = f"{a}, {b}"
```

```
for i in range(3, 11):
```

```
    a, b = b, a + b
```

```
    output += f", {b}"
```

```
print(output)
```

• [uwcs@hopper:~/git-good]\$ git log --color --oneline --graph --decorate --all

```
* 9e5bfb3 (main) Updated range and added README
```

```
* 58aad9b (HEAD) Added fibonacci program
```

Commit still exists, we've just moved back in time

- [uwcs@hopper:~/git-good]\$ git checkout main
Previous HEAD position was 58aad9b Added fibonacci program
Switched to branch 'main'
- [uwcs@hopper:~/git-good]\$ ls -a
. .. fibonacci.py .git README.md
- [uwcs@hopper:~/git-good]\$ █

Branches: into the multiverse

- We can make "alternative universes"





Branches: into the multiverse

- We can make "alternative universes"
- You can think of a branch as just a series of commits
- We've used branches already!
 - "main" (sometimes "master") is the default branch we started on
 - Generally kept both up-to-date and not broken...
 - Most workplace code has a “dev” and “production” branch

Why branches?

- Sometimes we want to experiment
 - If it doesn't work out, just discard the branch
- Helps isolate feature development
- Makes collaborative work easier (we'll discuss this more later)

How branches?

- The `git branch` command creates a new branch starting at the commit you're currently on
- To commit to the new branch, check it out!
- Modern version is `git switch <branch-name>`

- [uwcs@hopper:~/git-good]\$ git branch
* main
- [uwcs@hopper:~/git-good]\$ git branch user-input
- [uwcs@hopper:~/git-good]\$ git branch
* main
user-input
- [uwcs@hopper:~/git-good]\$ git switch user-input
Switched to branch 'user-input'
- [uwcs@hopper:~/git-good]\$ git branch
main
* user-input
- [uwcs@hopper:~/git-good]\$ █

 fibonacci.py

```
1  import sys
2
3  a = 0
4  b = 1
5
6  output = f"{a}, {b}"
7  terms = int(sys.argv[1])
8
9  for i in range(3, terms + 1):
10     a, b = b, a + b
11     output += f", {b}"
12
13  print(output)
```

- [uwcs@hopper:~/git-good]\$ python3 fibonacci.py 5
0, 1, 1, 2, 3
- [uwcs@hopper:~/git-good]\$ python3 fibonacci.py 10
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
- [uwcs@hopper:~/git-good]\$ git add --all
- [uwcs@hopper:~/git-good]\$ git commit -m "Added user input for number of terms"
[user-input fcd5ea] Added user input for number of terms
1 file changed, 4 insertions(+), 1 deletion(-)
- [uwcs@hopper:~/git-good]\$ █

```
⊗ [uwcs@hopper:~/git-good]$ python3 fibonacci.py we-have-a-bug
Traceback (most recent call last):
  File "/home/uwcs/git-good/fibonacci.py", line 7, in <module>
    terms = int(sys.argv[1])
            ^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'we-have-a-bug'

○ [uwcs@hopper:~/git-good]$ █
```

fibonacci.py

```
1  import sys
2
3  a = 0
4  b = 1
5
6  output = f"{a}, {b}"
7
8  try:
9      terms = int(sys.argv[1])
10 except:
11     print("Not a number, defaulting to 10")
12     terms = 10
13
14 for i in range(3, terms + 1):
15     a, b = b, a + b
16     output += f", {b}"
17
18 print(output)
```


- `[uwcs@hopper:~/git-good]$ python3 fibonacci.py bugs-begone`
Not a number, defaulting to 10
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
- `[uwcs@hopper:~/git-good]$ git add --all`
- `[uwcs@hopper:~/git-good]$ git commit -m "added input validation"`
[user-input 1e88108] added input validation
1 file changed, 6 insertions(+), 1 deletion(-)
- `[uwcs@hopper:~/git-good]$` █

Questions?

But what if we changed the main branch?

```
fibonacci.py
3
4   output = f"{a}, {b}"
5
6   for i in range(3, 16):
7       a, b = b, a + b
8       output += f", {b}"
9
10  print(f"The first fibonacci numbers are: {output}")
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

- [uwcs@hopper:~/git-good]\$ git switch main
Switched to branch 'main'
- [uwcs@hopper:~/git-good]\$ git add --all
- [uwcs@hopper:~/git-good]\$ git commit -m "added an output explanation"
[main 3e987f1] added an output explanation
1 file changed, 1 insertion(+), 1 deletion(-)
- [uwcs@hopper:~/git-good]\$ █

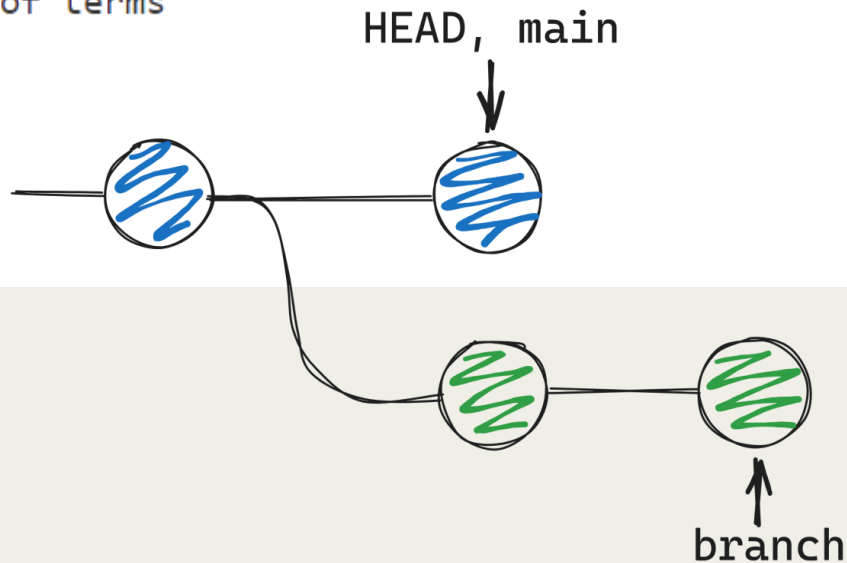
```
● [uwcs@hopper:~/git-good]$ git log --color --oneline --graph --decorate --all
* 3e987f1 (HEAD -> main) added an output explanation
| * 1e88108 (user-input) added input validation
| * fcdf5ea Added user input for number of terms
|/
* 9e5bfb3 Updated range and added README
* 58aad9b Added fibonacci program

○ [uwcs@hopper:~/git-good]$ █
```

This is hard to read....

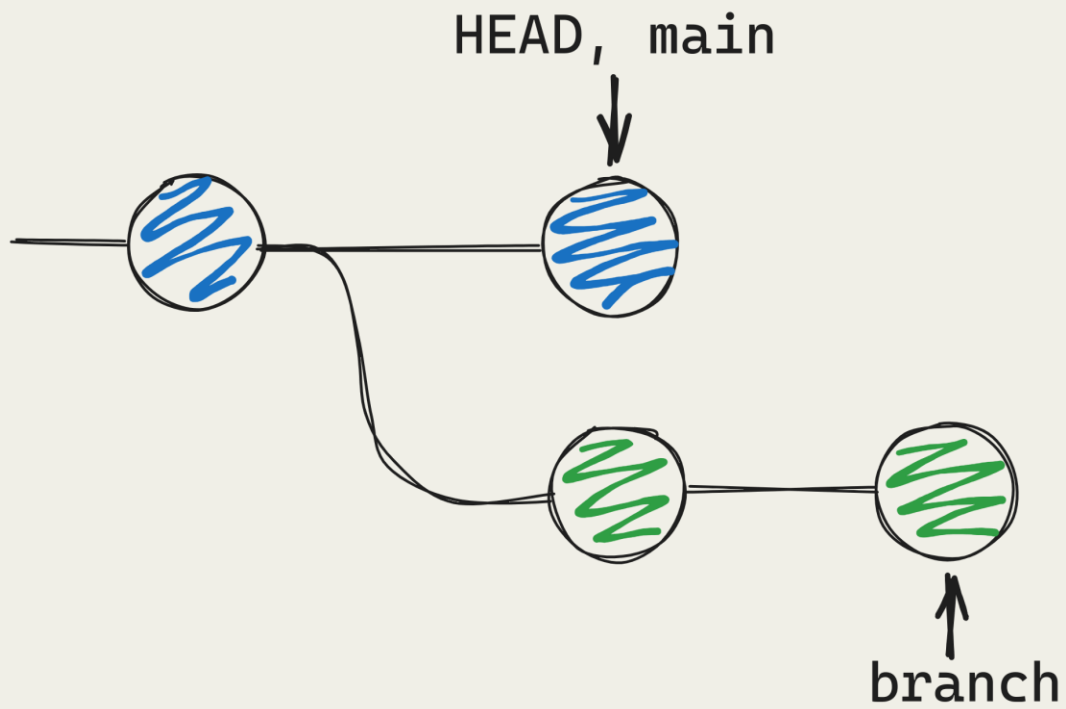
```
● [uwcs@hopper:~/git-good]$ git log --color --oneline --graph --decorate --all
* 3e987f1 (HEAD -> main) added an output explanation
| * 1e88108 (user-input) added input validation
| * fcdf5ea Added user input for number of terms
|/
* 9e5bfb3 Updated range and added README
* 58aad9b Added fibonacci program

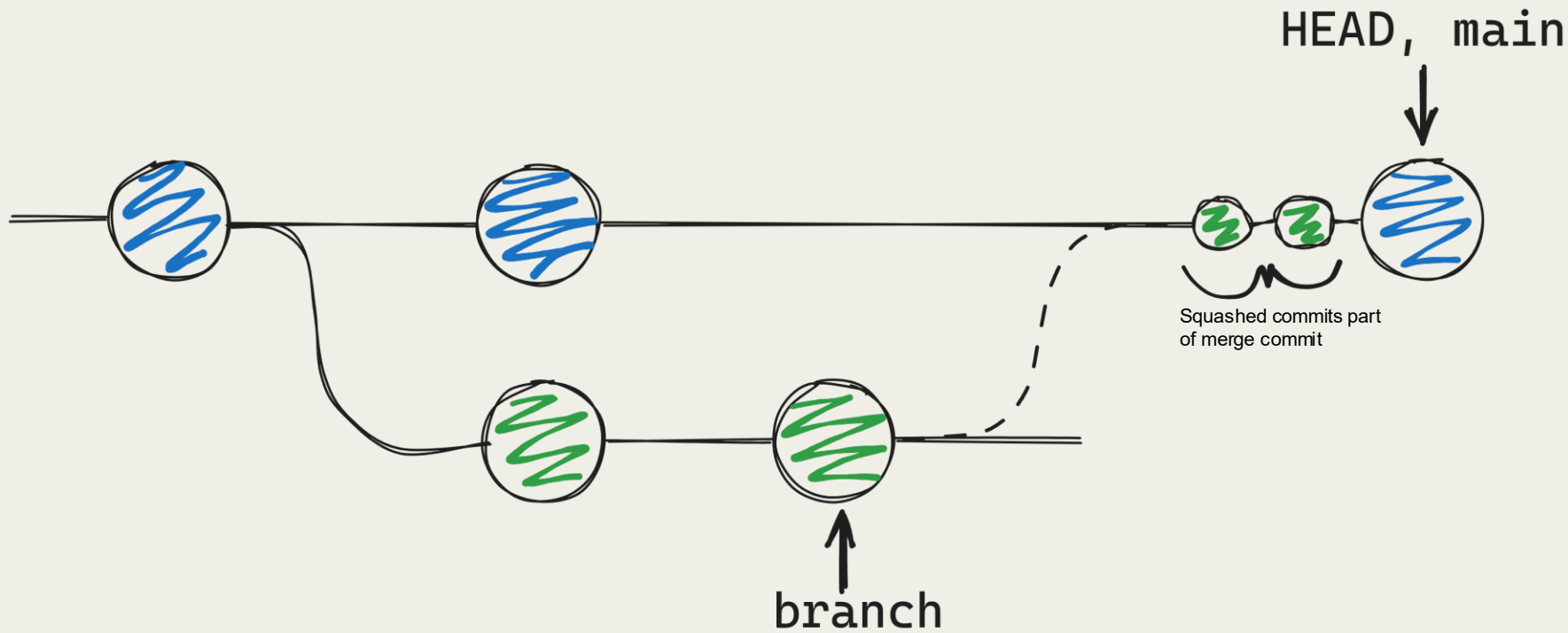
○ [uwcs@hopper:~/git-good]$
```

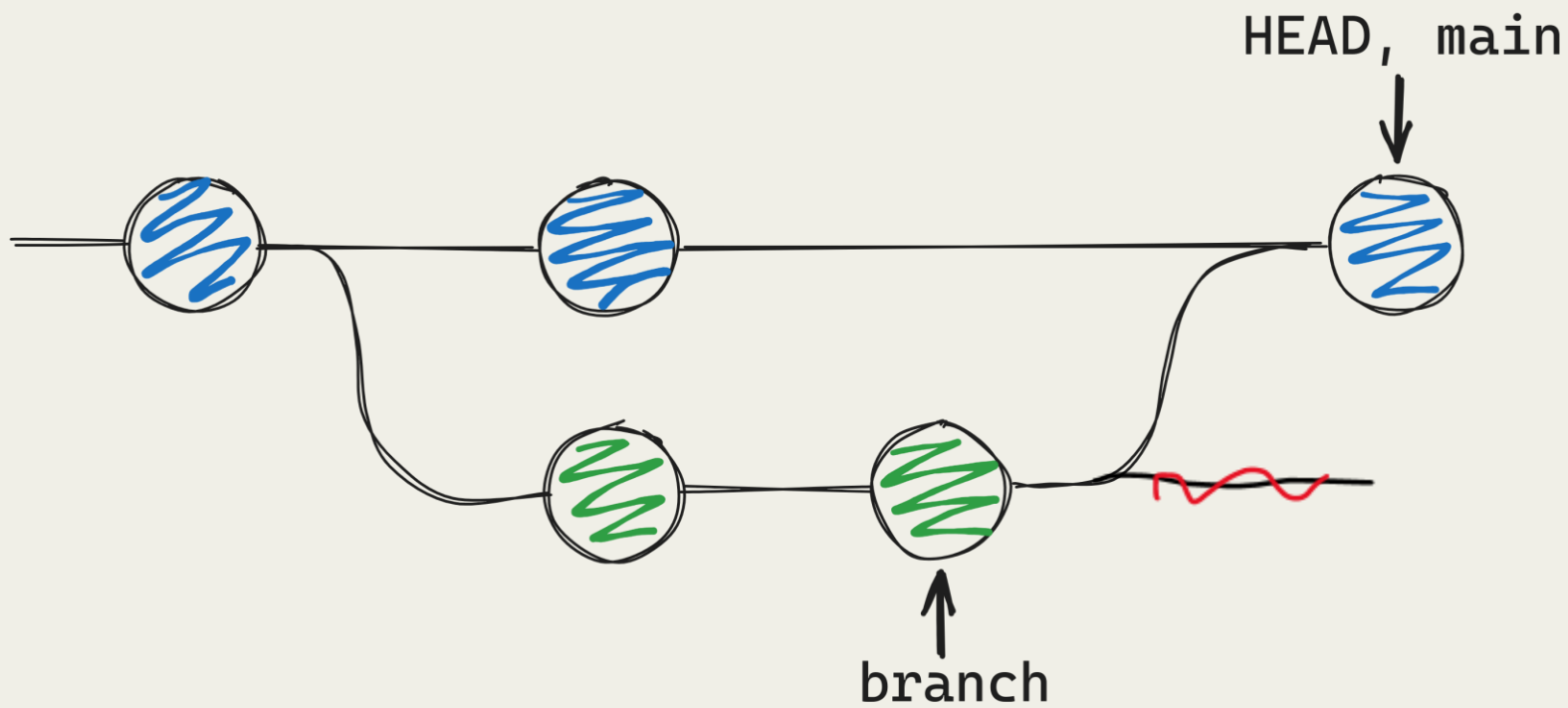


What is merging?

- Sometimes we want to have changes from more than one branch
 - For example, if we developed a feature in a branch, and want to include it in our main branch
- We can "merge" branch B into branch A to give branch A the changes from branch B







How to merge

- Switch to the branch you want to merge ***into***
- Use the `git merge <other>` command to merge the other branch into it
 - Creates a new commit containing the changes from the other branch on the current branch
 - Does not modify the other branch

```
[uwcs@hopper:~/git-good]$ git switch main  
Already on 'main'
```

- [uwcs@hopper:~/git-good]\$ git merge user-input

GNU nano 8.0

/home/uwcs/git-good/.git/MERGE_MSG

Merge branch 'user-input'

Please enter a commit message to explain why this merge is necessary,
especially if it merges an updated upstream into a topic branch.

Lines starting with '#' will be ignored, and an empty message aborts
the commit.

[Read 6 lines]

^G Help

^O Write Out

^F Where Is

^K Cut

^T Execute

^C Location

M-U Undo

M-A Set Mark

^X Exit

^R Read File

^ Replace

^U Paste

^J Justify

^/ Go To Line

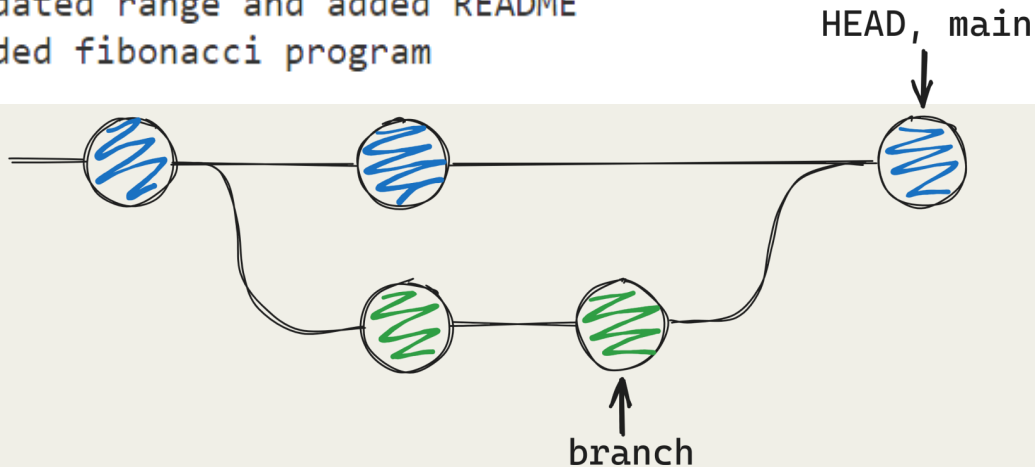
M-E Redo

M-6 Copy

```
[uwcs@hopper:~/git-good]$ git switch main  
Already on 'main'
```

- [uwcs@hopper:~/git-good]\$ git merge user-input
Auto-merging fibonacci.py
Merge made by the 'ort' strategy.
 fibonacci.py | 10 ++++++++-
 1 file changed, 9 insertions(+), 1 deletion(-)
- [uwcs@hopper:~/git-good]\$

```
● [uwcs@hopper:~/git-good]$ git log --color --oneline --graph --decorate --all
* 67b6c03 (HEAD -> main) Merge branch 'user-input'
| \
| * 1e88108 (user-input) added input validation
| * fcd5f5ea Added user input for number of terms
| * | 3e987f1 added an output explanation
| /
* 9e5bfb3 Updated range and added README
* 58aad9b Added fibonacci program
```



fibonacci.py

```
1  import sys
2
3  a = 0
4  b = 1
5
6  output = f"{a}, {b}"
7
8  try:
9      terms = int(sys.argv[1])
10 except:
11     print("Not a number, defaulting to 10")
12     terms = 10
13
14 for i in range(3, terms + 1):
15     a, b = b, a + b
16     output += f", {b}"
17
18 print(f"The first fibonacci numbers are: {output}")
```

Something broke my merge!

- Sometimes, git is unable to merge automatically
 - For example, a line is changed in both branches, and it doesn't know which one to pick
- This is called a merge conflict
- You have to resolve it manually
 - Outside the scope of this course, but lots of online tutorials
- **You should be careful when doing this**

Questions?

Changing history



- One of the benefits of version control is easily fixing mistakes!
- The `git revert` command undoes a single commit
 - Creates a new commit doing the undoing the old one
- The `git reset` command is more dangerous
 - Won't discuss now, look it up if you need it
 - Soft/mixed doesn't affect working directory, only HEAD
 - Hard **discards everything** back to a specified point

fibonacci.py

```
1  import sys
2
3  a = 0
4  b = 1
5
6  print("Garden tiger moth")
7
8  output = f"{a}, {b}"
9
10 try:
11     terms = int(sys.argv[1])
12 except:
13     print("Not a number, defaulting to 10")
14     terms = 10
15
16 for i in range(3, terms + 1):
17     a, b = b, a + b
18     output += f", {b}"
19
20 print(f"The first fibonacci numbers are: {output}")
```

- [uwcs@hopper:~/git-good]\$ git add --all
- [uwcs@hopper:~/git-good]\$ git commit -m "bugged code"
[main ea0386b] bugged code
1 file changed, 2 insertions(+)
- [uwcs@hopper:~/git-good]\$ git log --color --oneline --graph --decorate --all
* ea0386b (HEAD -> main) bugged code
* 67b6c03 Merge branch 'user-input'
|\
| * 1e88108 (user-input) added input validation
| * fcdf5ea Added user input for number of terms
* | 3e987f1 added an output explanation
|/
* 9e5bfb3 Updated range and added README
* 58aad9b Added fibonacci program
- [uwcs@hopper:~/git-good]\$ █

- `[uwcs@hopper:~/git-good]$ git revert ea0386b`
`[main b4867ed] Revert "bugged code"`
`1 file changed, 2 deletions(-)`

GNU nano 8.0

/home/uwcs/git-good/.git/COMMIT_EDITMSG

Revert "bugged code"

This reverts commit ea0386bb4e9058ccceea5ee336116f3957b7678e.

Please enter the commit message for your changes. Lines starting
with '#' will be ignored, and an empty message aborts the commit.

On branch main
Changes to be committed:
modified: fibonacci.py
#

[Read 11 lines]

^G Help

^O Write Out

^F Where Is

^K Cut

^T Execute

^C Location

M-U Undo

M-A Set Mark

^X Exit

^R Read File

^ Replace

^U Paste

^J Justify

^/ Go To Line

M-E Redo

M-6 Copy

```
* b4867ed Revert "bugged code"
* ea0386b bugged code
* 67b6c03 Merge branch 'user-input'
|\
| * 1e88108 added input validation
| * fcdf5ea Added user input for number of terms
* | 3e987f1 added an output explanation
|/
* 9e5bfb3 Updated range and added README
* 58aad9b Added fibonacci program
```

Remote work

- Up to now git has been all local
- 🌟 internet 🌟
- Remote repos are versions of a repo that live online
- The `git remote` command lets you manage them
- This allows us to collaborate!
- GitHub, GitLab, and others offer remote repo hosting
 - You can also do it yourself for a challenge!

But first... a secret challenger approaches

- From August 2021, boring ol' password auth to github.com became no longer possible
- Now you have 2 options:
 - Authentication tokens
 - SSH keys
- Fear not, for we have full guide for this over at: <https://uwcs.co.uk/resources/github-token-authentication/>
- (Raven hopes their slightly half-arsed tweaks are enough, poke them if you *really* want a slides-based tutorial too lol)

For ssh key enjoyers:

`git@github.com:UWCS/git-good-demo.git`

- `[uwcs@hopper:~/git-good]$ git remote add origin https://github.com/UWCS/git-good-demo.git`
- `[uwcs@hopper:~/git-good]$ git push origin main`
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 16 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (23/23), 2.30 KiB | 588.00 KiB/s, done.
Total 23 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), done.
To https://github.com/UWCS/git-good-demo.git
* [new branch] main -> main
- `[uwcs@hopper:~/git-good]$` █



git-good-demo

Public



Edit Pins



Unwatch 4



Fork 0



Star 0



main



1 Branch



0 Tags



Go to file



Add file



Code



uwcsgithub Revert "bugged code"

b4867ed · 22 minutes ago

8 Commits



README.md

Updated range and added README

yesterday



fibonacci.py

Revert "bugged code"

22 minutes ago



README



Git Good Demo

This program prints out Fibonacci numbers

This was written for UWCS' Git Good talk

About



No description, website, or topics provided.



Readme



Activity



Custom properties



0 stars



4 watching



0 forks

Report repository

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Remotes and cloning

- You can get a local copy of a remote repo by "cloning" it
- The `git clone` subcommand does this
- Sometimes software is distributed by cloning the repo, then building/running it yourself
 - Called building from source

- [uwcs@hopper:~/git-good]\$ ls -a
.
..

For ssh key enjoyers:

git@github.com:UWCS/git-good-demo.git

- [uwcs@hopper:~/git-good]\$ git clone https://github.com/UWCS/git-good-demo
Cloning into 'git-good-demo'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 23 (delta 6), reused 23 (delta 6), pack-reused 0 (from 0)
Receiving objects: 100% (23/23), done.
Resolving deltas: 100% (6/6), done.

- [uwcs@hopper:~/git-good]\$ ls -a
.
.. git-good-demo

- [uwcs@hopper:~/git-good]\$ ls -a git-good-demo/
.
.. fibonacci.py .git README.md

- [uwcs@hopper:~/git-good]\$ █

Remotes and branches

- Local branches can correspond to remote branches
 - The remote copy is called `<remote>/<branch>`, for example `origin/main`
 - You can have local branches which aren't on the remote (and vice versa)

Fetch, Push, and Pull

- `git fetch` updates what the local repo knows about the remote repo
- `git push` updates the remote branch from the local branch
- `git pull` updates the local branch from the remote branch
 - This is like a `git fetch` followed by a `git merge <remote>/<branch>`



git-good-demo /

README.md

in **main**

Cancel changes

Commit changes...

Edit

Preview

Spaces ▾

2 ▾

Soft wrap ▾

```
1  # Git Good Demo
2
3  This program prints out Fibonacci numbers
4
5  This was written for UWCS' Git Good talk
6
7  GitHub also allows you to edit via a web interface|
```


Commit changes



Commit message

Update README.md

Extended description

Add an optional extended description..

- ☒ Commit directly to the main branch
- ☐ Create a **new branch** for this commit and start a pull request

Cancel

Commit changes

Fetching changes

- Often your IDE of choice will have a git extension that will (sometimes) auto-magically pull changes
- If doing manually this can be accomplished by:
 - `git fetch origin` (this is what IDEs will automatically run)
 - Fetches changes down but doesn't update HEAD
 - `git pull origin main`
 - Pulls changes down and “fast-forward” the HEAD to most recent remote

```

• [uwcs@hopper:~/git-good/git-good-demo]$ git log --color --oneline --graph --decorate --all
* d535385 (origin/main, origin/HEAD) Update README.md
* b4867ed (HEAD -> main) Revert "bugged code"
* ea0386b bugged code
* 67b6c03 Merge branch 'user-input'
| \
| * 1e88108 added input validation
| * fcdf5ea Added user input for number of terms
* | 3e987f1 added an output explanation
| /
* 9e5bfb3 Updated range and added README
* 58aad9b Added fibonacci program

```

The result of git fetch origin

```

• [uwcs@hopper:~/git-good/git-good-demo]$ git log --color --oneline --graph --decorate --all
* d535385 (origin/main, origin/HEAD) Update README.md
* b4867ed (HEAD -> main) Revert "bugged code"
* ea0386b bugged code
* 67b6c03 Merge branch 'user-input'
| \
| * 1e88108 added input validation
| * fcdf5ea Added user input for number of terms
* | 3e987f1 added an output explanation
| /
* 9e5bfb3 Updated range and added README
* 58aad9b Added fibonacci program

```

The result of git pull origin main

git-good-demo > ⓘ README.md > abc # Git Good Demo

1 **# Git Good Demo**

2

3 This program prints out Fibonacci numbers

4

5 This was written for UWCS' Git Good talk

6


7 GitHub also allows you to edit via a web interface

8

9 You can still edit locally|

10

- `[uwcs@hopper:~/git-good/git-good-demo]$ git add --all`
- `[uwcs@hopper:~/git-good/git-good-demo]$ git commit -m "Updated README again"`
`[main c6163b9] Updated README again`
`1 file changed, 2 insertions(+)`
- `[uwcs@hopper:~/git-good/git-good-demo]$ git push origin main`
`Enumerating objects: 5, done.`
`Counting objects: 100% (5/5), done.`
`Delta compression using up to 16 threads`
`Compressing objects: 100% (3/3), done.`
`Writing objects: 100% (3/3), 339 bytes | 339.00 KiB/s, done.`
`Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)`
`remote: Resolving deltas: 100% (1/1), completed with 1 local object.`
`To https://github.com/UWCS/git-good-demo`
`d535385..c6163b9 main -> main`
- `[uwcs@hopper:~/git-good/git-good-demo]$` █

 uwcsgithub Updated README again

c6163b9 · 1 minute ago  History

Preview

Code

Blame

9 lines (5 loc) · 181 Bytes

Raw



Git Good Demo

This program prints out Fibonacci numbers

This was written for UWCS' Git Good talk

GitHub also allows you to edit via a web interface

You can still edit locally

Recap so far

- Looking at histories with `git log`
- Travelling in time with `git checkout`
- Working with branches with `git branch/merge`
- Undoing mistakes with `git revert`
- Working remotely with `git remote/...`

Top tips

- DO NOT COMMIT SECRETS
- Commit little and often
- Give commits meaningful names
- Make small branches and merge regularly
- Clean up dead branches
 - Can be done with `git branch -d <branch name>`

💫 ***Do not commit secrets!!!*** 💫

Installation

- Windows: <https://git-scm.com/download/win>
- Mac: <https://git-scm.com/download/mac>
- Both come with options to just use the command line or to download a GUI program as well
- For Linux, it is almost always pre-installed (otherwise use a package manager of your choice)

Hate the command line?

- Lots of software exists to help manage git repos graphically
 - Git GUI for windows
 - SourceTree for Mac
- Almost all modern IDEs also have git plugins
 - This includes VSCode!
 - This is how you will most likely use git

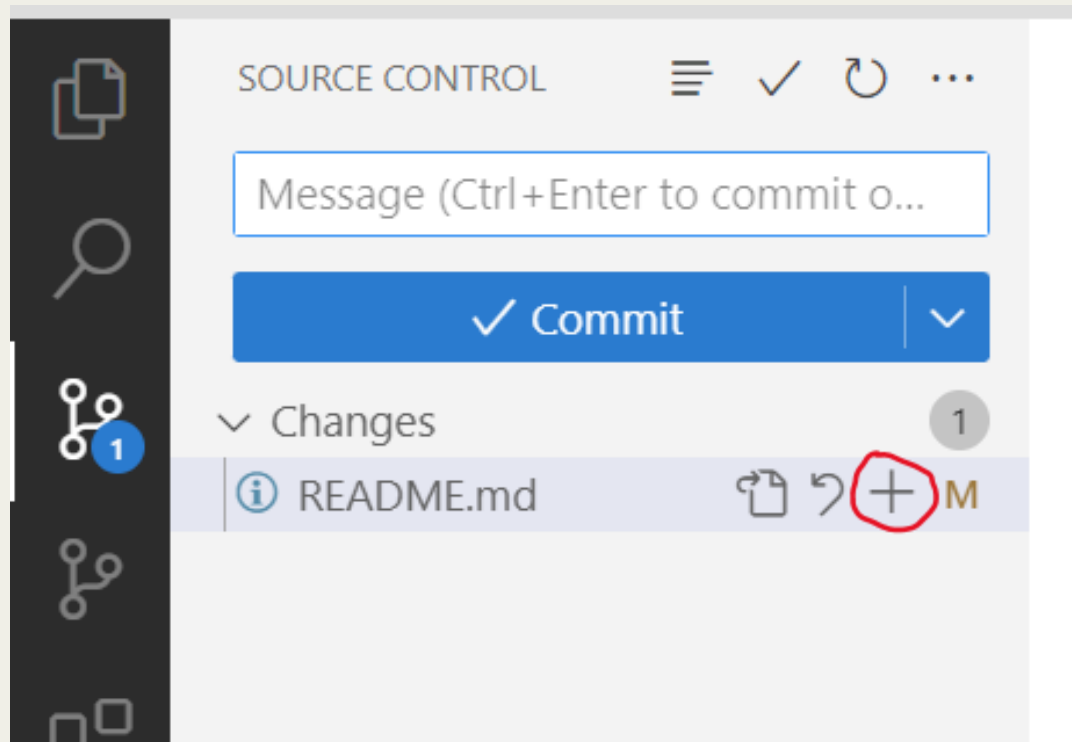
9 You can still edit locally

10

11

12

This was edited using the VSCode extension





SOURCE CONTROL



updated README for final time

✓ Commit



✓ Staged Changes

1

ⓘ README.md

M



✓ Changes

0

SOURCE CONTROL



Message (Ctrl+Enter to commit o...

 Sync Changes 1 ↑



Preview

Code

Blame

11 lines (6 loc) · 225 Bytes

Raw



Git Good Demo

This program prints out Fibonacci numbers

This was written for UWCS' Git Good talk

GitHub also allows you to edit via a web interface

You can still edit locally

This was edited using the VSCode extension





File Edit Selection View Go Run ...



SOURCE CONTROL



- updated README for final ti...  
- Updated README again uwcs
- Update README.md uwcsgithub
- Revert "bugged code" uwcs
- bugged code uwcs
- Merge branch 'user-input' uwcs
 - added input validation uwcs
 - Added user input for number of t...
- added an output explanation uwcs
- Updated range and added READ...
- Added fibonacci program uwcs

Ignoring files

- Sometimes you don't want to keep track of certain files
 - Generated files (`.jar`, `__pycache__`), databases, etc.
 - Put secrets in an ignored `.env` file
- Create a file called `.gitignore` in the repository
 - This can contain a list of [globs](#) of filenames to ignore

Configuration

- Git is very configurable!
- Many things can be changed, including
 - Default editor, commit template, global gitignore, merge tool, aliases, handling of whitespace, default login credentials...
- Use the `git config` command to do this
 - Can do this on a project, user, or system level

🔥 EVERYTHING IS ON FIRE HELP 🔥

- Especially when inexperienced, it can be easy to mess up
- Someone has messed up exactly how you have before
- <https://ohshitgit.com> to fix many common mistakes

This was just an introduction

- We have barely scratched the surface of what git can do
 - Hopefully enough to get started/convince you git is useful




I want to learn more!

- Git Reference – <http://git.github.io/git-reference/index.html>
- Pro Git – <https://book.git-scm.com/book/en/v2>
- Learn Git Branching – <https://learngitbranching.js.org/>
- GitHub and Atlassian both have helpful pages on many topics

When will I ever use this???

- Good for programming course-work (e.g. 118)
- Eases collaboration in group projects
- When you get a job, it will probably use git in some way

I want to practice using git!

- Luckily for you, we are running a workshop!
 - Put everything we've covered today into practice
 - Get help if you get stuck
- During Comp Café next week
 - 5-7pm, Friday 4th October
 - TBC (probably the lab on the right as you come into DCS)
 - Free food!
 - Be there or be  !



**University of Warwick
Computing Society**

I am an (un-)paid shill...

- Hopefully you found this talk interesting!
 - If you did, we do loads more academic events throughout the year
 - If you are currently bored to death/asleep, we do other things too!



University of Warwick
Computing Society



<https://uwcs.co.uk/events/>

Questions?