Collaboration and Version Control with Git

Adnan Fiaz

Data Scientist

✉ afiaz@mango-solutions.com

@tapundemek

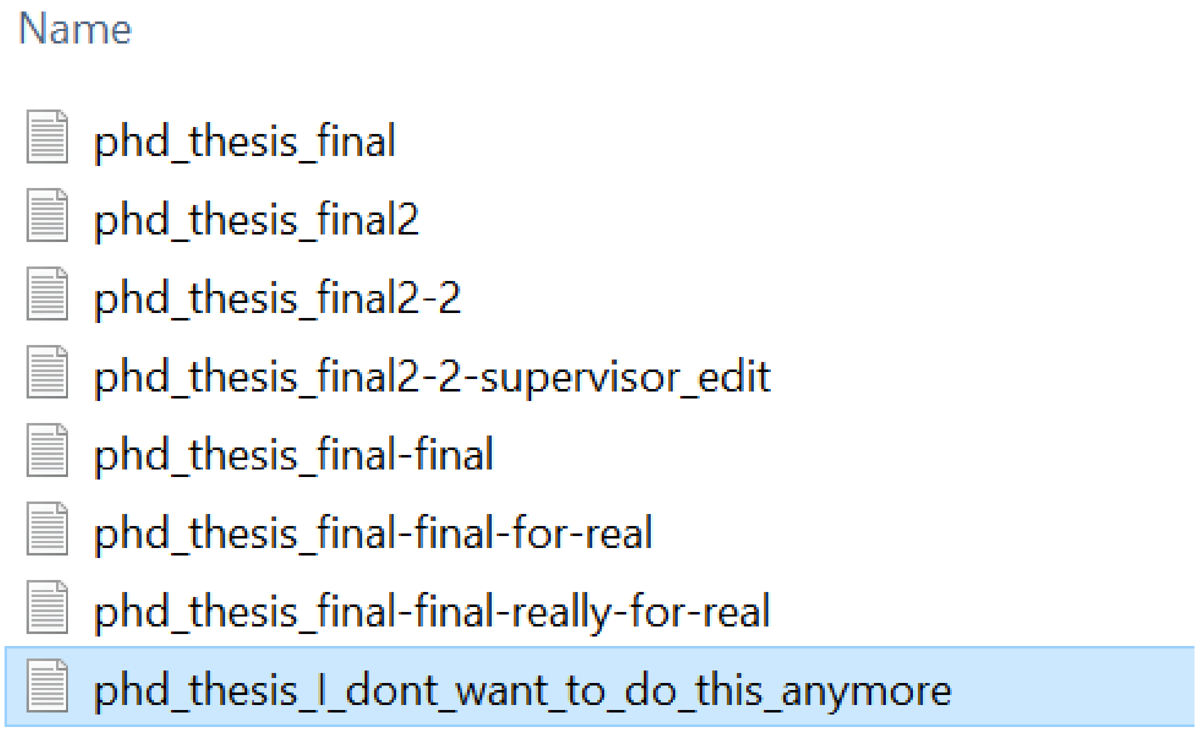
Agenda

- Version Control
- Git (local)
- Undoing Changes
- Branches
- Remotes
- Collaboration

Why Version Control?

Why use version control?

- Have your files ever looked anything like this?


Name

phd_thesis_final
phd_thesis_final2
phd_thesis_final2-2
phd_thesis_final2-2-supervisor_edit
phd_thesis_final-final
phd_thesis_final-final-for-real
phd_thesis_final-final-really-for-real
phd_thesis_I_dont_want_to_do_this_anymore

Why use version control?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Pro git, by Scott Chacon and Ben Straub

https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Why use version control?

- Other benefits:
  - Work on new features in a safe isolated environment (branches)
  - Greater ease of collaboration on code
  - Contributing to open-source projects

Why Git?

- Originally developed by Linus Torvalds, author of the Linux kernel
- Allegedly named this after himself as well
- Better and easier solution
  - Subversion
  - SourceSafe

The Local Git

Ways to use Git: CLI vs GUI

- Many graphical Git clients:
  - Source Tree
  - GitHub Desktop
  - TortoiseGit

- However…
  - All require configuration and customization

- CLI is universal – nothing you cannot do

Basic Terminal Usage

- Change directory: **`cd`**
  - Used alone moves to 'home' directory
    - Do this when using Git Bash on Windows as it can start in the **`root`** directory
  - Up 1 directory level: **`cd ..`**
  - Up 2 directory levels: **`cd ../..`**
- Make directory: **`mkdir`**
- List files in directory: **`ls`**
- Remove a file: **`rm`**

Initial Git Configuration:

- Type the following commands into the terminal:

```
$ git config --global user.name "James Bond"
$ git config --global user.email jbond@mi6.gov.uk
$ git config --global core.autocrlf true # Windows only
```

- Note that all git commands are prefixed with the word: **`git`**

Time to start a Git repo!

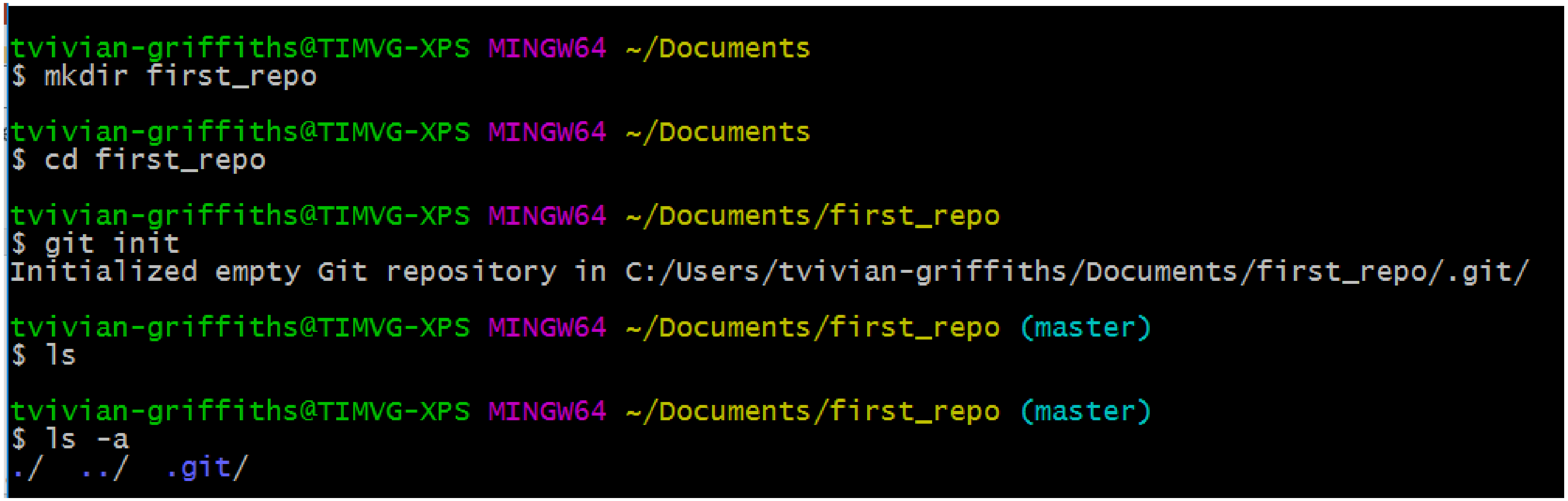
- Navigate to an available directory **`(cd)`**
  - I recommend **`Documents`**
  - Make a new directory: **`first_repo (mkdir)`**
  - Change into this new directory **`(cd)`**
- Then use the command to initialise a repo:

```
$ git init
```

- Add a text file to the directory **`(touch)`**

Git directory

- ‘Hidden’ directory – DON’T TOUCH THIS!
- Note the name: **`master`**

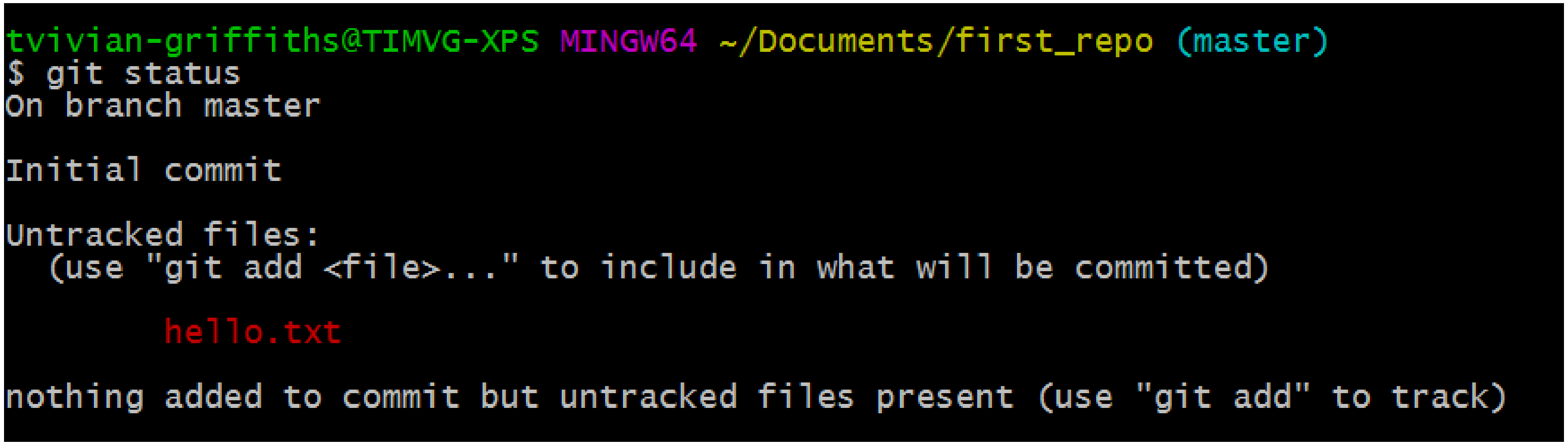


```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents
$ mkdir first_repo

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents
$ cd first_repo

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo
$ git init
Initialized empty Git repository in C:/Users/tvivian-griffiths/Documents/first_repo/.git/

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ ls

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ ls -a
./  ../  .git/
```

Checking the status of the repo

- Probably the most useful command in git:

```
$ git status
```

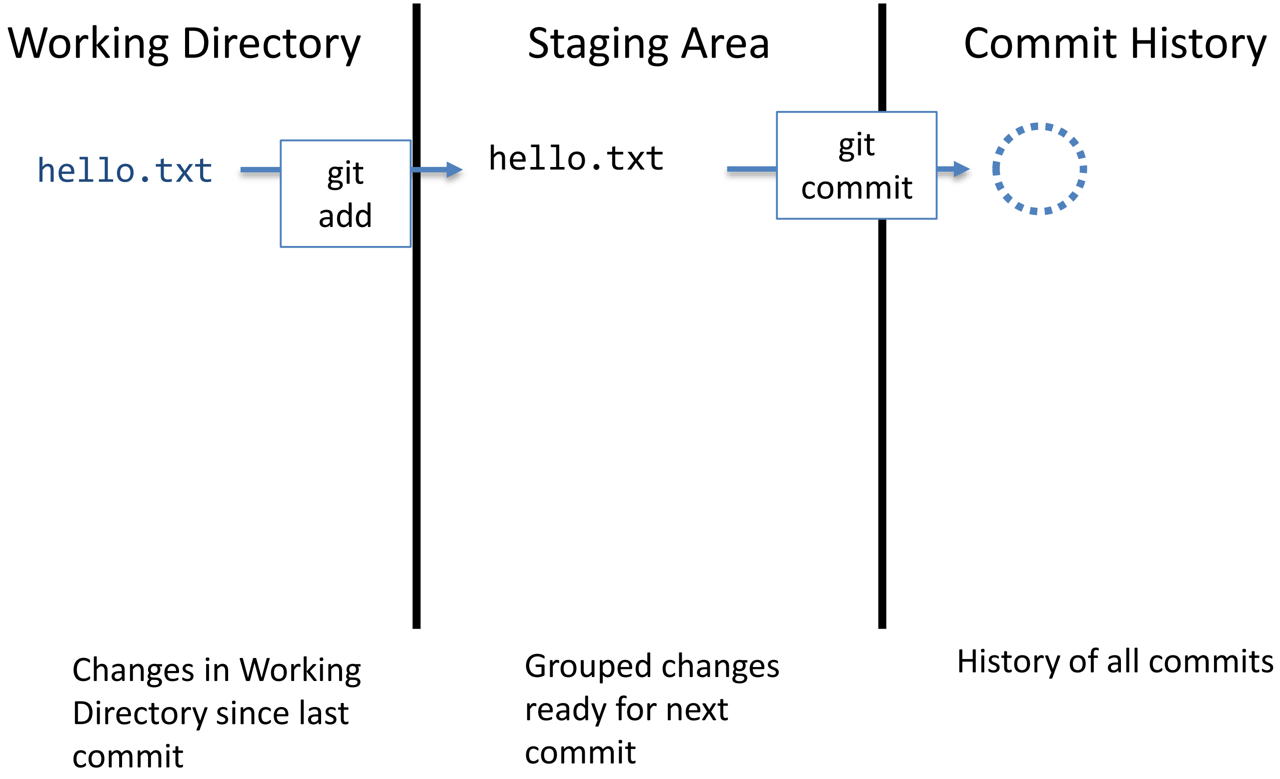


```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git status
On branch master

Initial commit

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

        hello.txt

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

Untracked files

- Files that are currently ‘unknown’ to Git
- Newly created files
- Once they have been added, they are then ‘tracked’ by Git

Basic Git Workflow



Basic Git Workflow

- **`hello.txt`** is currently in the Working directory
- We **`add`** it to the Staging Area
- We **`commit`** it to the Local Repository
- Drill these 2 words into your mind!!

Staging Area?? Why?

- The Staging Area allows us to choose what to commit if we have already edited many files
- We can also ‘top-up’ what is in the Staging Area, as **only** the changes in the Staging Area get committed
- A single commit should contain related changes for a common purpose

Add First File

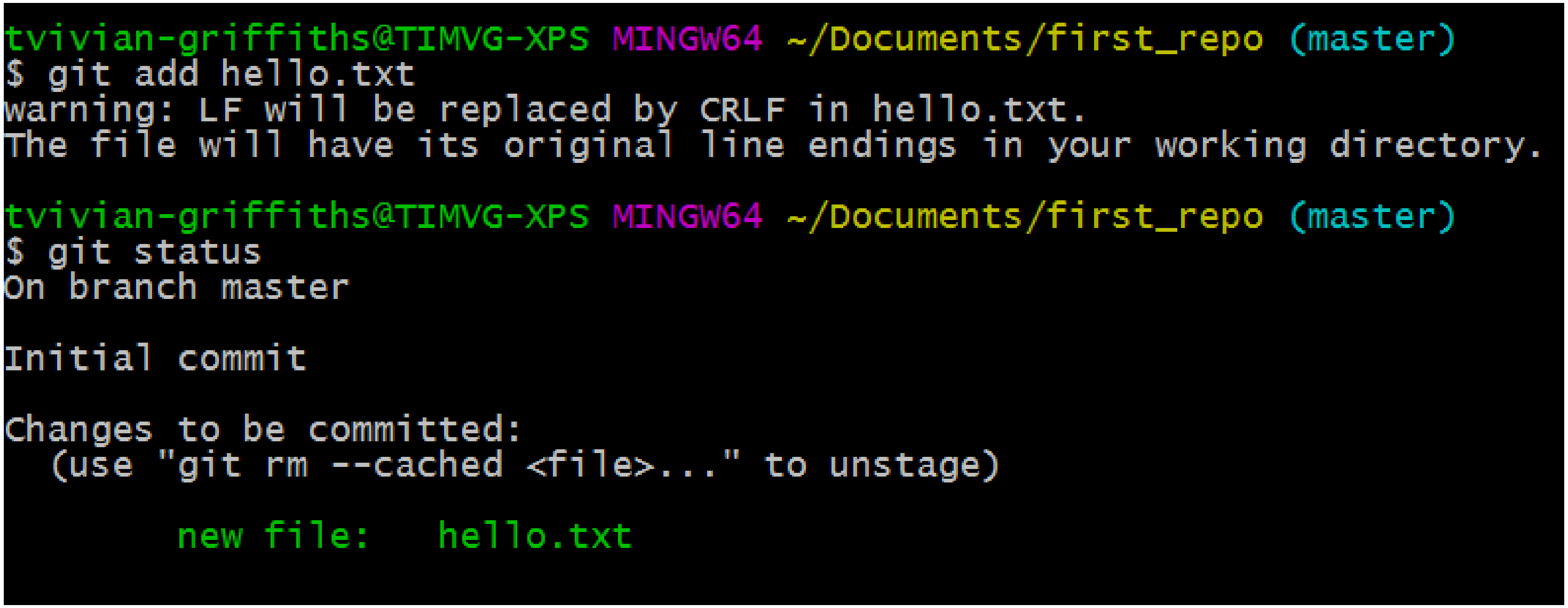
- Stage an untracked file (hello.txt) with

```
$ git add hello.txt
```

- View status

```
$ git status
```

Add First File



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git add hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory.

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git status
On branch master

Initial commit

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

        new file:   hello.txt
```

Commit First File

- Commit the changes (then write a message)

```
$ git commit
```

- Or for a short message

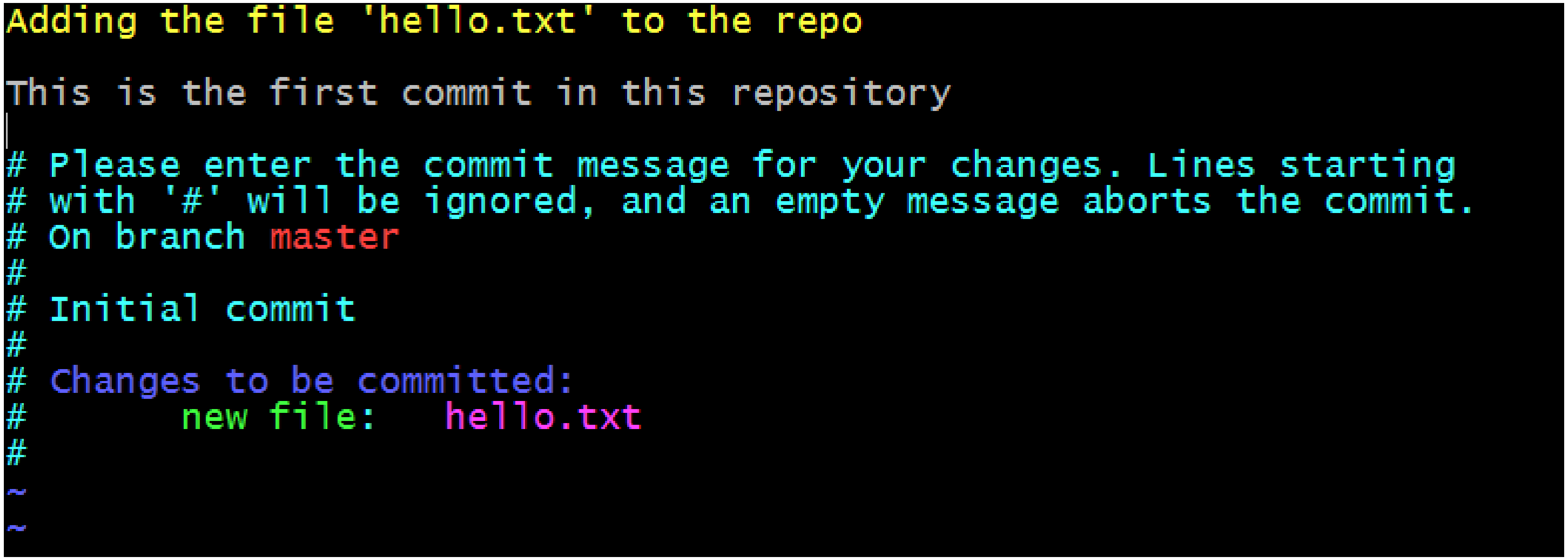
```
$ git commit -m "short message"
```

- View status

```
$ git status
```

Commit First File

- **`$ git commit`** – This will open VIM
- Then add the following lines – then **`:wq`**



```
Adding the file 'hello.txt' to the repo

This is the first commit in this repository

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   hello.txt
#
~
~
```

"There will be a day when I learn how to really use VIM properly"

Basic VIM usage

- Open file for editing as follows:

```
$ vim helloworld.txt
```

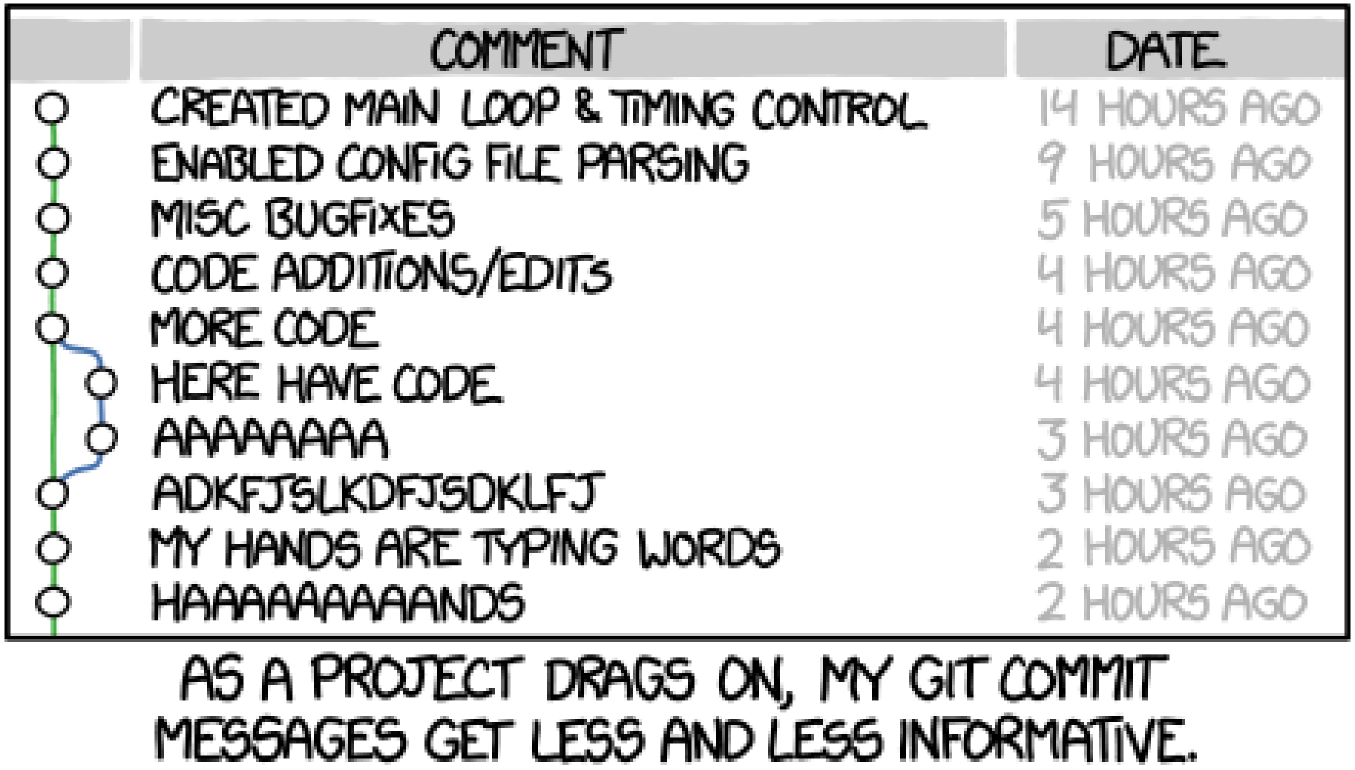
- Toggle between INSERT and NORMAL mode:
  - `i`
  - `ESC`
- Use arrow keys for navigation
  - Not really the 'VIM way' but… whatever…
- Save file **`(:w)`**, close file **`(:q)`**
- Save and close **`(:wq)`**, close no save **`(:q!)`**

Replace VIM with notepad

- Since Git 2.5.3

```
$ git config core.editor notepad
```

Good and Bad 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

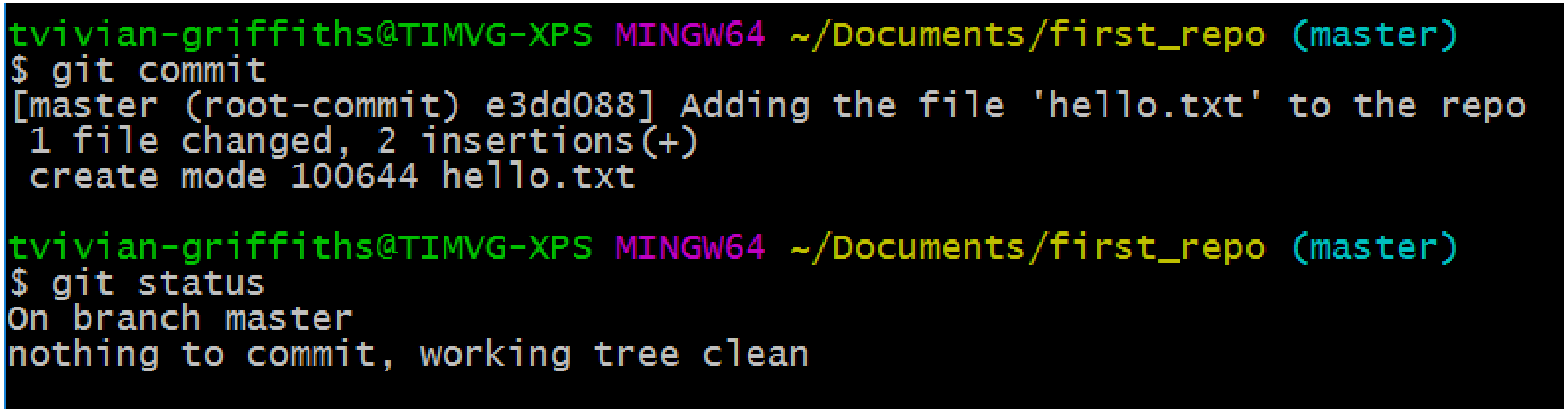
AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Source: xkcd at https://xkcd.com/1296/

Writing Good Commit Messages

- First line <50 characters, concise summary. Complete the sentence:
  - "If the changes are added they will <this bit should be your commit message first line>
- Then empty line, and more detailed discussion
- Wrap text at ~72 characters
- Refer to issues, related commits, users involved
- Try answering the questions:
  - Why is this change necessary?
  - How does it address the issue?
  - What side effects does this change have?

Check the new status



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git commit
[master (root-commit) e3dd088] Adding the file 'hello.txt' to the repo
 1 file changed, 2 insertions(+)
 create mode 100644 hello.txt

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git status
On branch master
nothing to commit, working tree clean
```

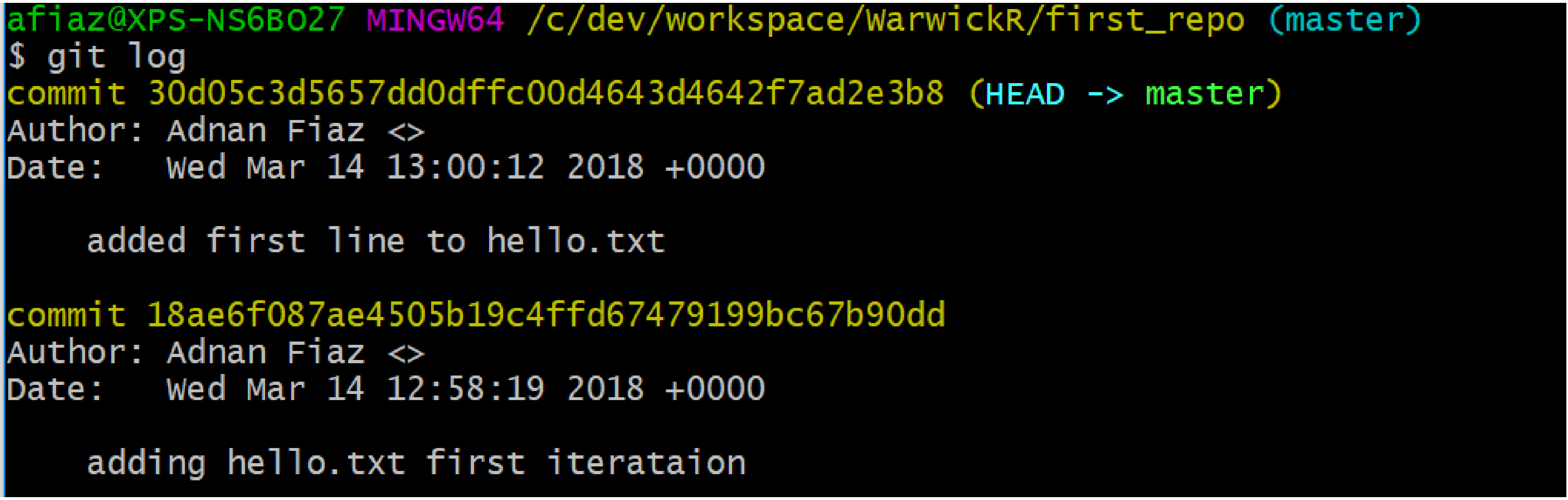
Modify a file

- Edit your text file. Then run a status

```
$ git status
```

- The file is now registered as modified (M)
- Add the changes to the staging area
- Commit the changes to the repo

Viewing previous commits: `git log`



```
afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git log
commit 30d05c3d5657dd0dffc00d4643d4642f7ad2e3b8 (HEAD -> master)
Author: Adnan Fiaz <>
Date:   Wed Mar 14 13:00:12 2018 +0000

    added first line to hello.txt

commit 18ae6f087ae4505b19c4ffd67479199bc67b90dd
Author: Adnan Fiaz <>
Date:   Wed Mar 14 12:58:19 2018 +0000

    adding hello.txt first iteratation
```

- Note – this opens a ‘pager’ so you can scroll through the text
- Use any key to scroll and ‘q’ to exit
- Or use **git log --oneline –decorate --graph --all**

Key Points

- Git represents changes in files as a series of checkpoints know as “commits”
- You can control what changes to include in each commit using the staging area.
- A single commit should contain related changes for a common purpose
- Committing little and often is highly encouraged!

Undoing changes

Traveling through time...

- Git as a timeline management utility
- Amending existing commit

  `git commit --amend`
- Undoing committed changes:

  `git revert` and **`git reset`**
- Undoing uncommitted changes:

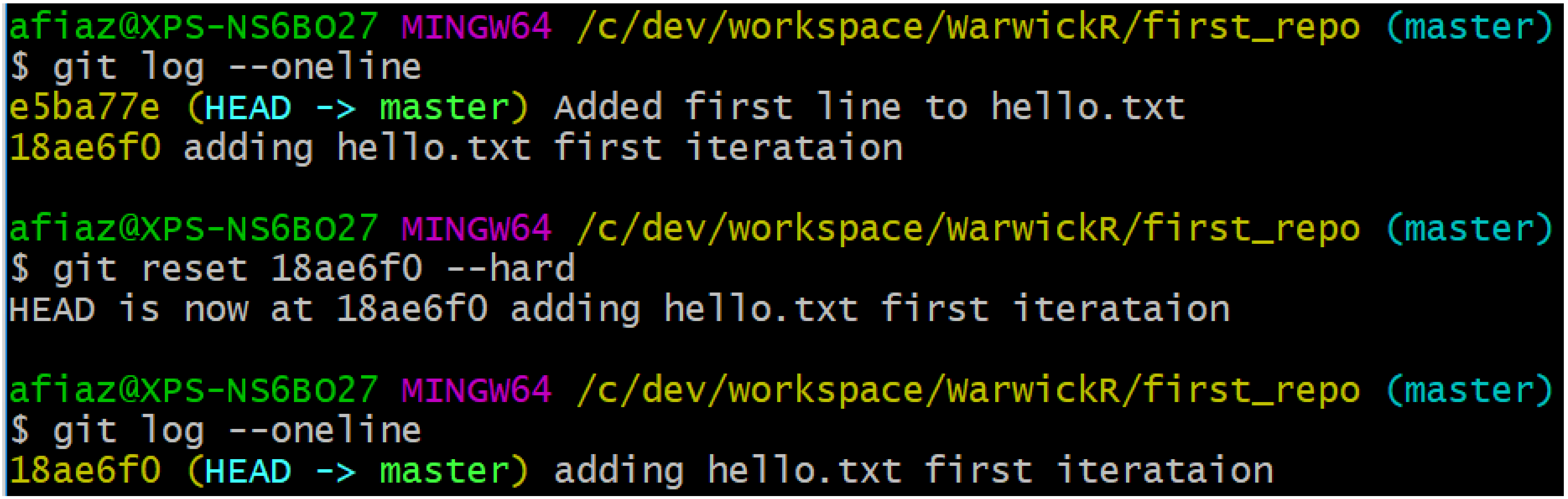
  `git reset` and **`git checkout`**

Undoing committed changes

git reset

- **`git reset`** removes commits from the history
- Only appropriate for local repo's (as we will see)

Undoing committed changes
git reset



```
afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git log --oneline
e5ba77e (HEAD -> master) Added first line to hello.txt
18ae6f0 adding hello.txt first iterataion

afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git reset 18ae6f0 --hard
HEAD is now at 18ae6f0 adding hello.txt first iterataion

afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git log --oneline
18ae6f0 (HEAD -> master) adding hello.txt first iterataion
```

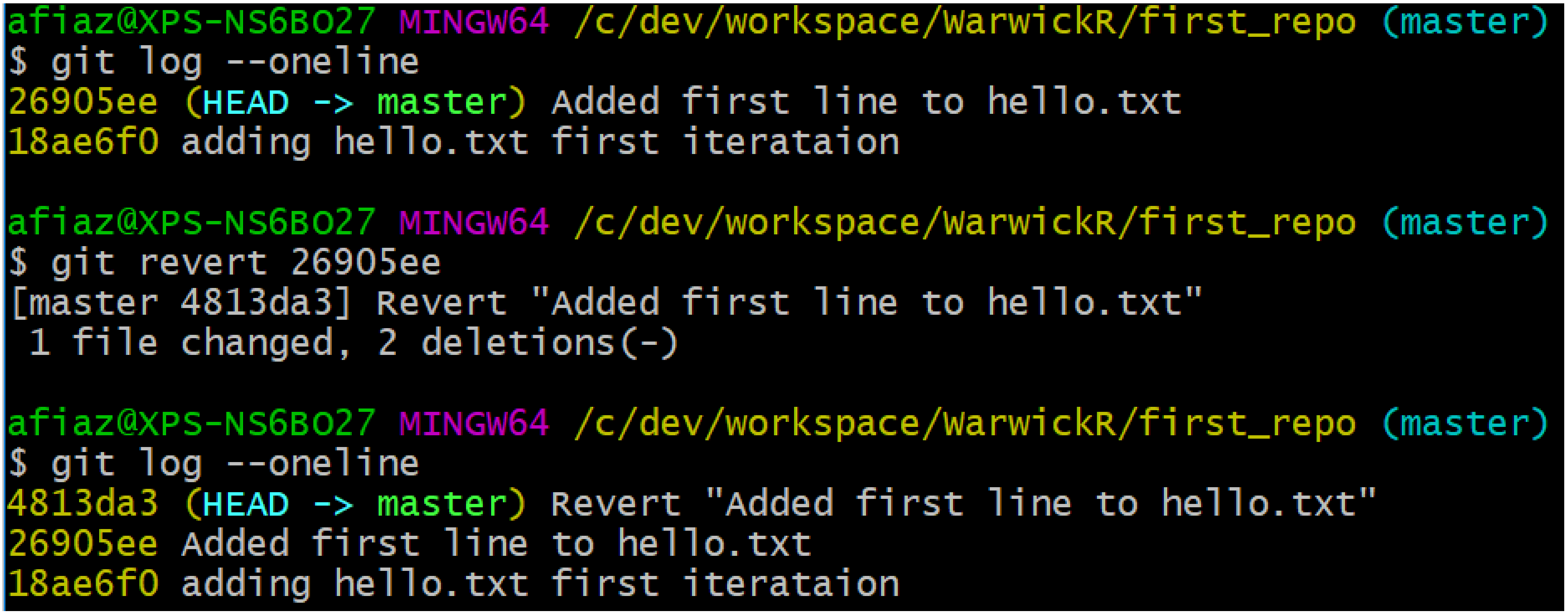
Undoing committed changes

git revert

- **`git revert`** undoes a commit by creating a new commit
- Considered the safe approach for public changes

Undoing committed changes

git revert



```
afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git log --oneline
26905ee (HEAD -> master) Added first line to hello.txt
18ae6f0 adding hello.txt first iteratAion

afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git revert 26905ee
[master 4813da3] Revert "Added first line to hello.txt"
 1 file changed, 2 deletions(-)

afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git log --oneline
4813da3 (HEAD -> master) Revert "Added first line to hello.txt"
26905ee Added first line to hello.txt
18ae6f0 adding hello.txt first iteratAion
```

Undoing uncommitted changes

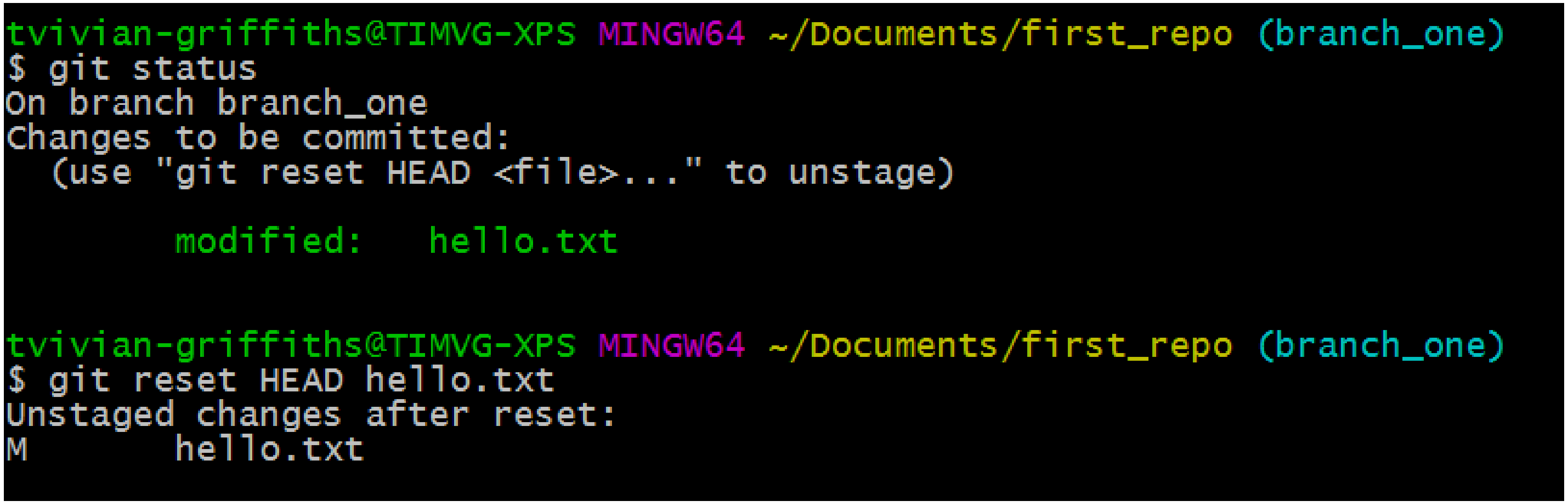
- Undo changes to Staging Area:

```
$ git reset HEAD hello.txt
```

- Undo changes to Working Directory

```
$ git checkout -- hello.txt
```

Removing from Staging Area



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ git status
On branch branch_one
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   hello.txt


tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ git reset HEAD hello.txt
Unstaged changes after reset:
M       hello.txt
```

Undoing uncommitted changes

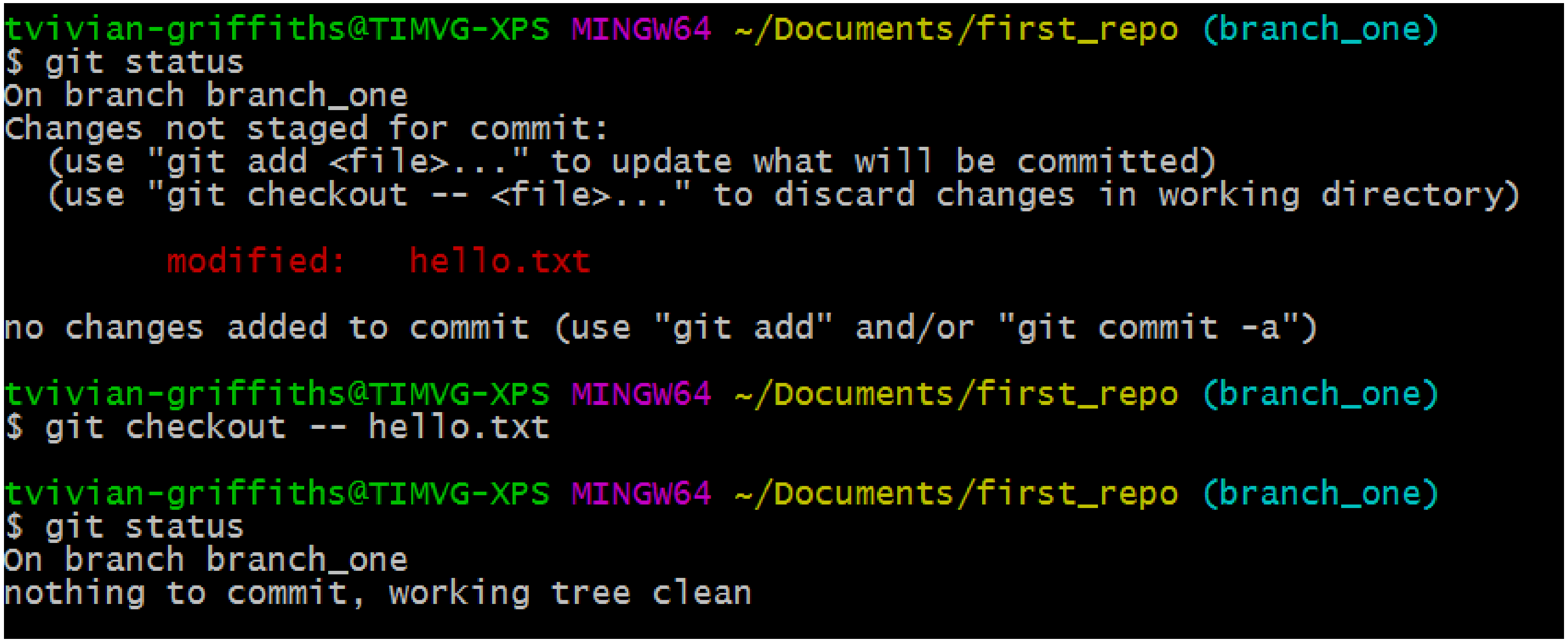
- Undo changes to Staging Area:

```
$ git reset HEAD hello.txt
```

- Undo changes to Working Directory

```
$ git checkout -- hello.txt
```

Undo changes in Working Directory



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ git status
On branch branch_one
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:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ git checkout -- hello.txt

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ git status
On branch branch_one
nothing to commit, working tree clean
```

Detached Head

- You can **`git checkout`** any commit
- This doesn't affect the commit history
- However, you will no longer be referring to the *master* branch

- Welcome to the 'detached HEAD' state!
- Try it out!

Removing files

- Don't delete files without telling git first. Use:

```
$ git rm <file>
```

- Or if renaming:

```
$ git mv <oldfile> <newfile>
```

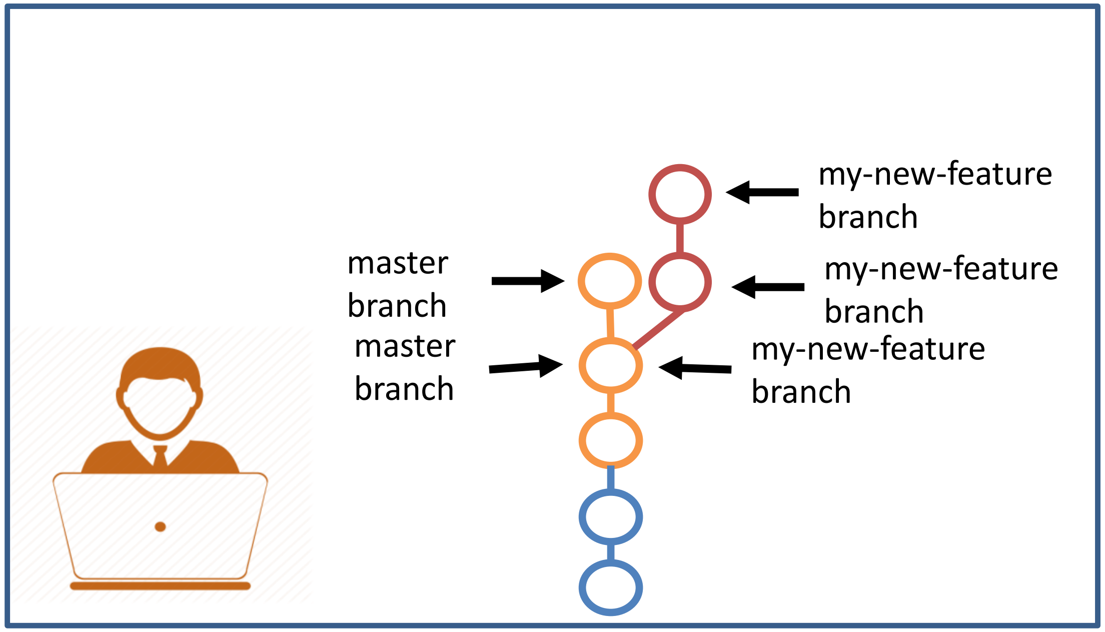
- This will do the move, and stage the changes

Intermezzo / Break

- http://starlogs.net/#Selbosh/scrooge
- http://www.commitlogsfromlastnight.com/

Branches

Branching



- “Branches” are pointers to commits in the history.
- Default is called “master”
- Can easily add more.
- Each can be progressed independently
- Useful for quick experimentation

Create and move to a new branch

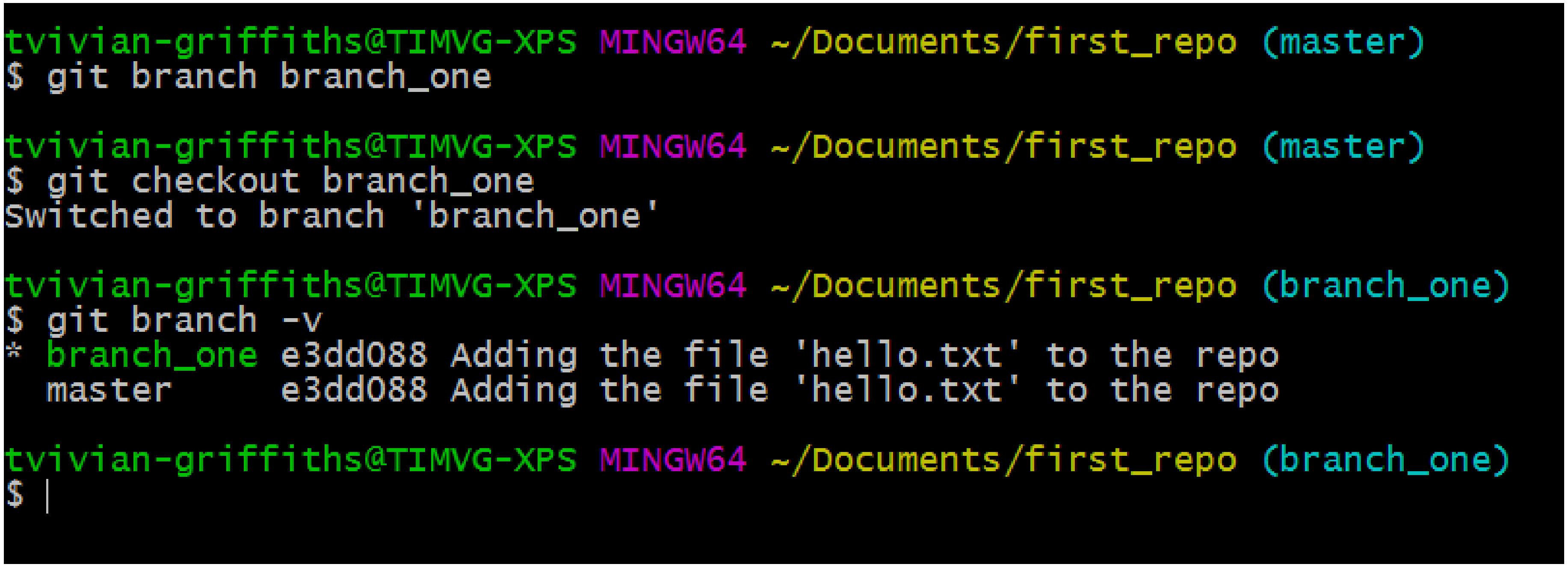
- Run the following commands to create, and move focus to a branch called **`branch_one`**

```
$ git branch branch_one
$ git checkout branch_one
```

- We can view all existing branches as follows:

```
$ git branch -v
```

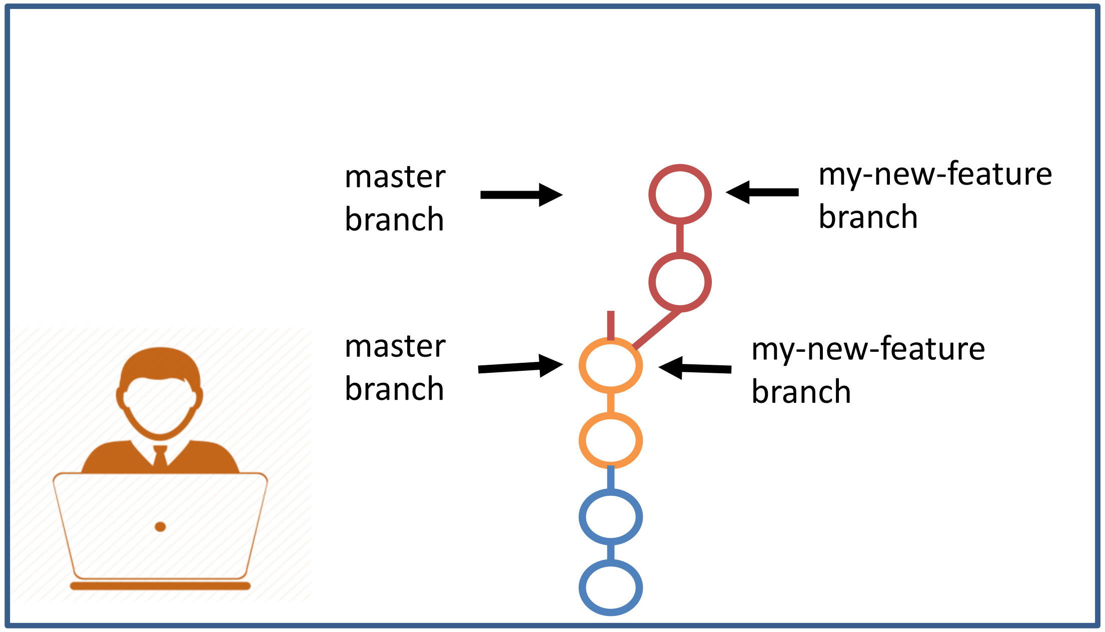
Create and move to a new branch


```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git branch branch_one

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git checkout branch_one
Switched to branch 'branch_one'

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ git branch -v
* branch_one e3dd088 Adding the file 'hello.txt' to the repo
  master     e3dd088 Adding the file 'hello.txt' to the repo

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ |
```

Exercise

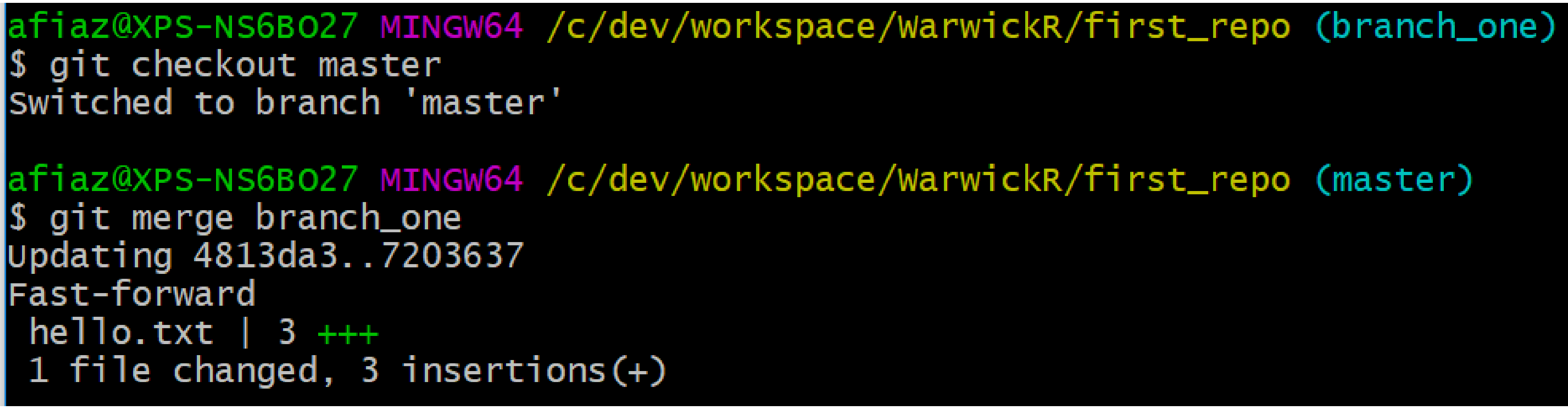
- Do the following in this new branch
- Perform a **`$ git status`** between each step
  1. Edit the current **`hello.txt`** file
  2. Add the changes to the staging area
  3. Commit the changes to the repo

Merging – Fast Forward



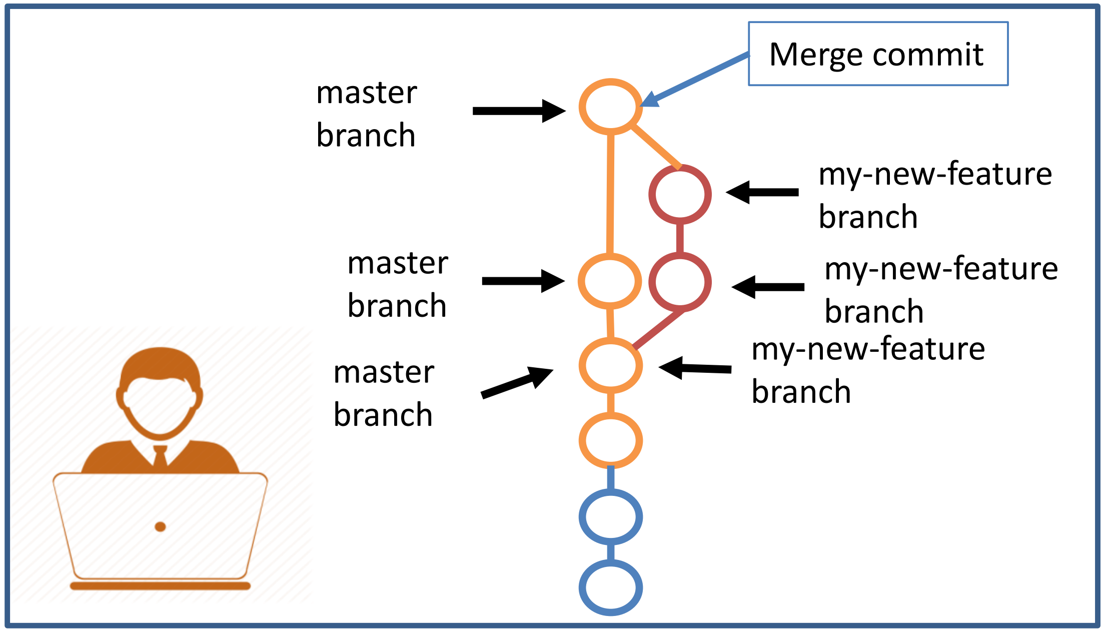
- If nothing has changed on the master, git can "fast forward"
- Replays your changes on the target branch

Merging changes back to **`master`**



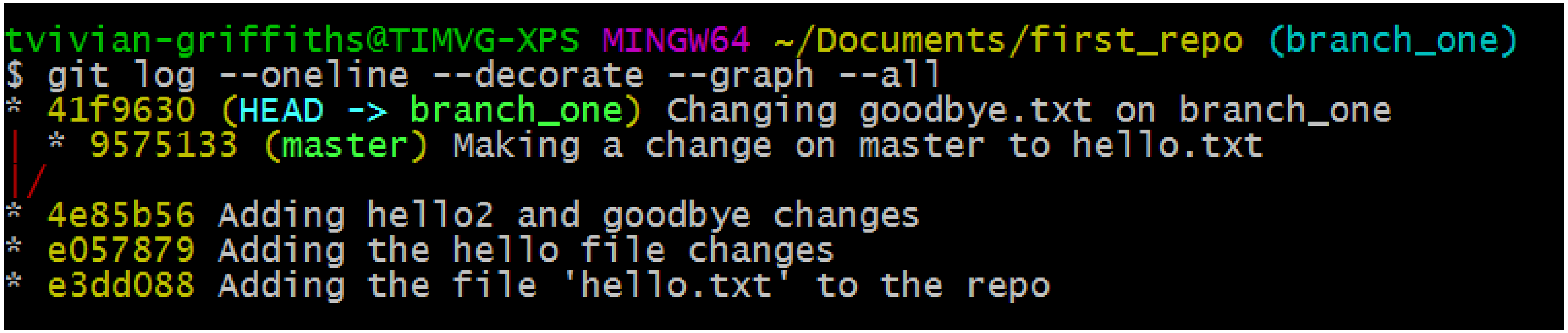
```
afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (branch_one)
$ git checkout master
Switched to branch 'master'

afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git merge branch_one
Updating 4813da3..7203637
Fast-forward
 hello.txt | 3 +++
 1 file changed, 3 insertions(+)
```

Merging – Changes on master



- Git can automatically merge where it's clear what to do

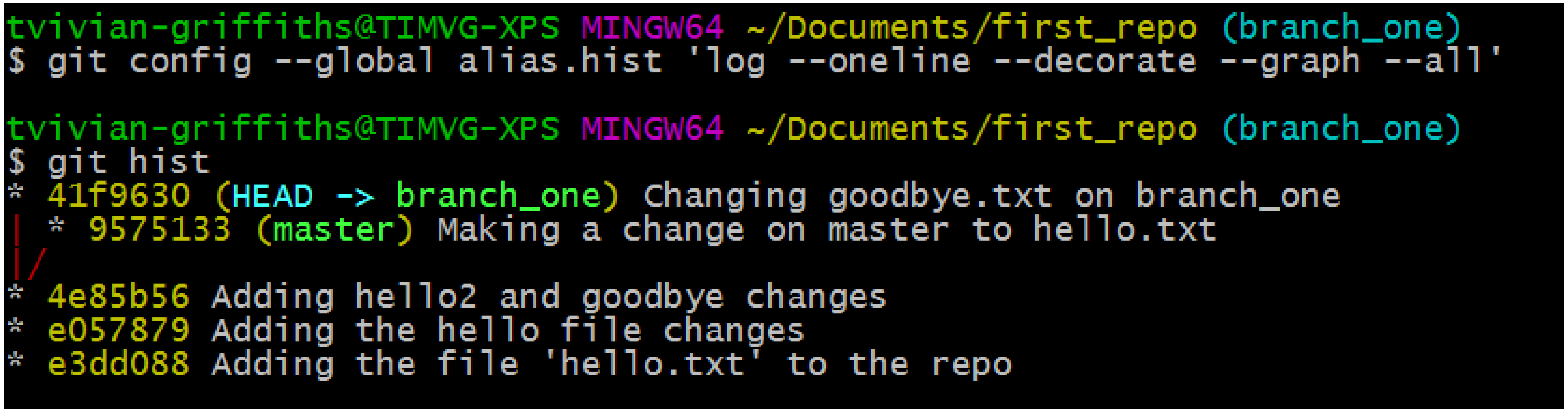
A very useful log method



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ git log --oneline --decorate --graph --all
* 41f9630 (HEAD -> branch_one) Changing goodbye.txt on branch_one
| * 9575133 (master) Making a change on master to hello.txt
|/

* 4e85b56 Adding hello2 and goodbye changes
* e057879 Adding the hello file changes
* e3dd088 Adding the file 'hello.txt' to the repo
```

- Note that this method shows how the branches have diverged
- The `--all` tag is used to show later commits if you have moved focus to an earlier branch
- `HEAD ->` shows where the current focus is

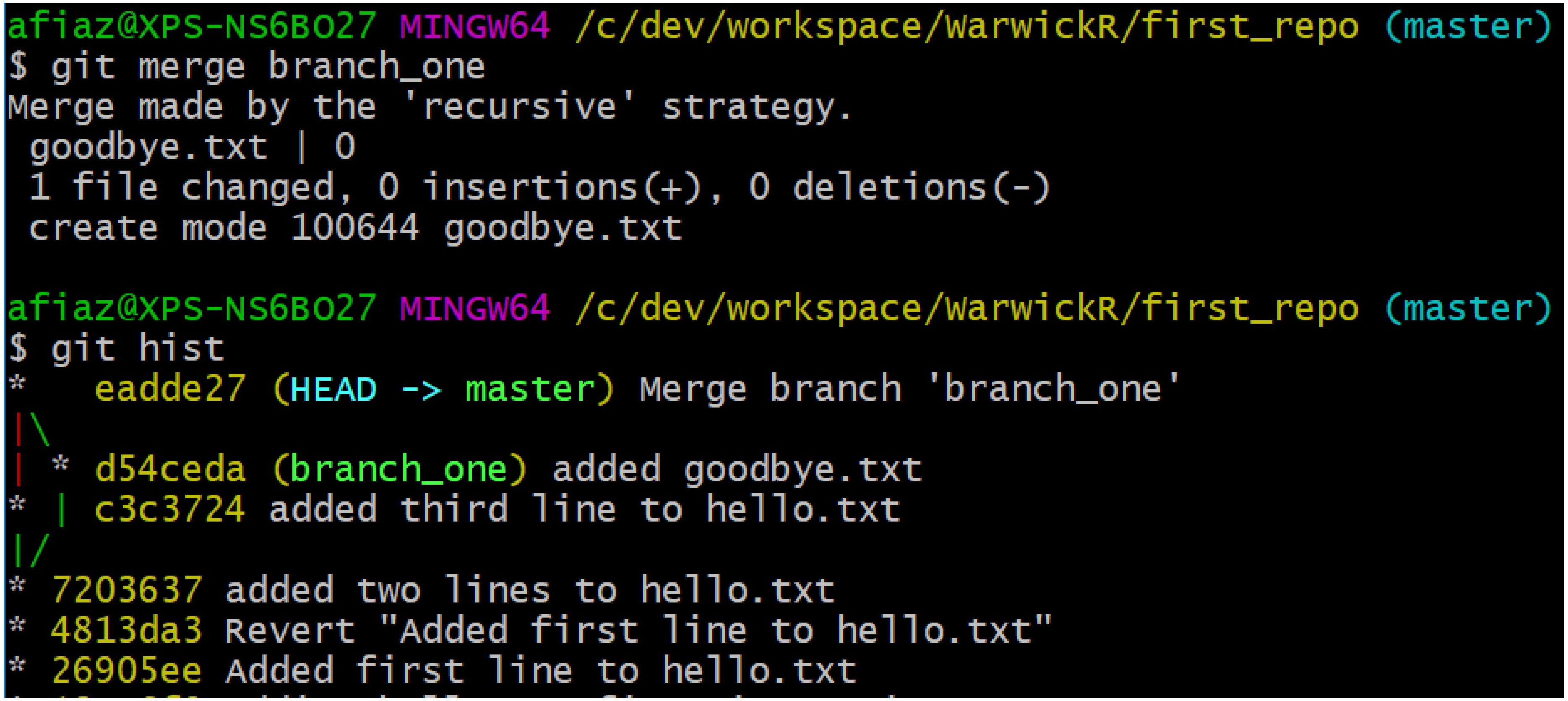
Making an alias command



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ git config --global alias.hist 'log --oneline --decorate --graph --all'

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_one)
$ git hist
* 41f9630 (HEAD -> branch_one) Changing goodbye.txt on branch_one
| * 9575133 (master) Making a change on master to hello.txt
|/
* 4e85b56 Adding hello2 and goodbye changes
* e057879 Adding the hello file changes
* e3dd088 Adding the file 'hello.txt' to the repo
```

- Credit to Jason Taylor for this

Automatic merge

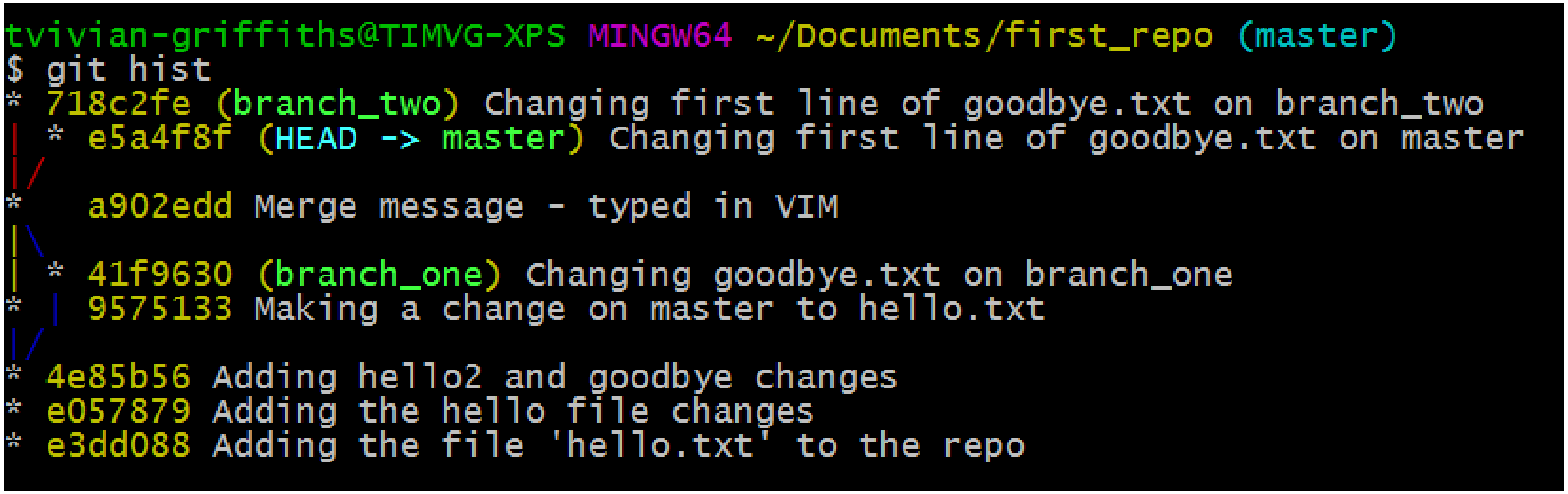


```
afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git merge branch_one
Merge made by the 'recursive' strategy.
 goodbye.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 goodbye.txt

afiaz@XPS-NS6BO27 MINGW64 /c/dev/workspace/WarwickR/first_repo (master)
$ git hist
*   eadde27 (HEAD -> master) Merge branch 'branch_one'
|\
| * d54ceda (branch_one) added goodbye.txt
* | c3c3724 added third line to hello.txt
|/
* 7203637 added two lines to hello.txt
* 4813da3 Revert "Added first line to hello.txt"
* 26905ee Added first line to hello.txt
```

When merges go bad!

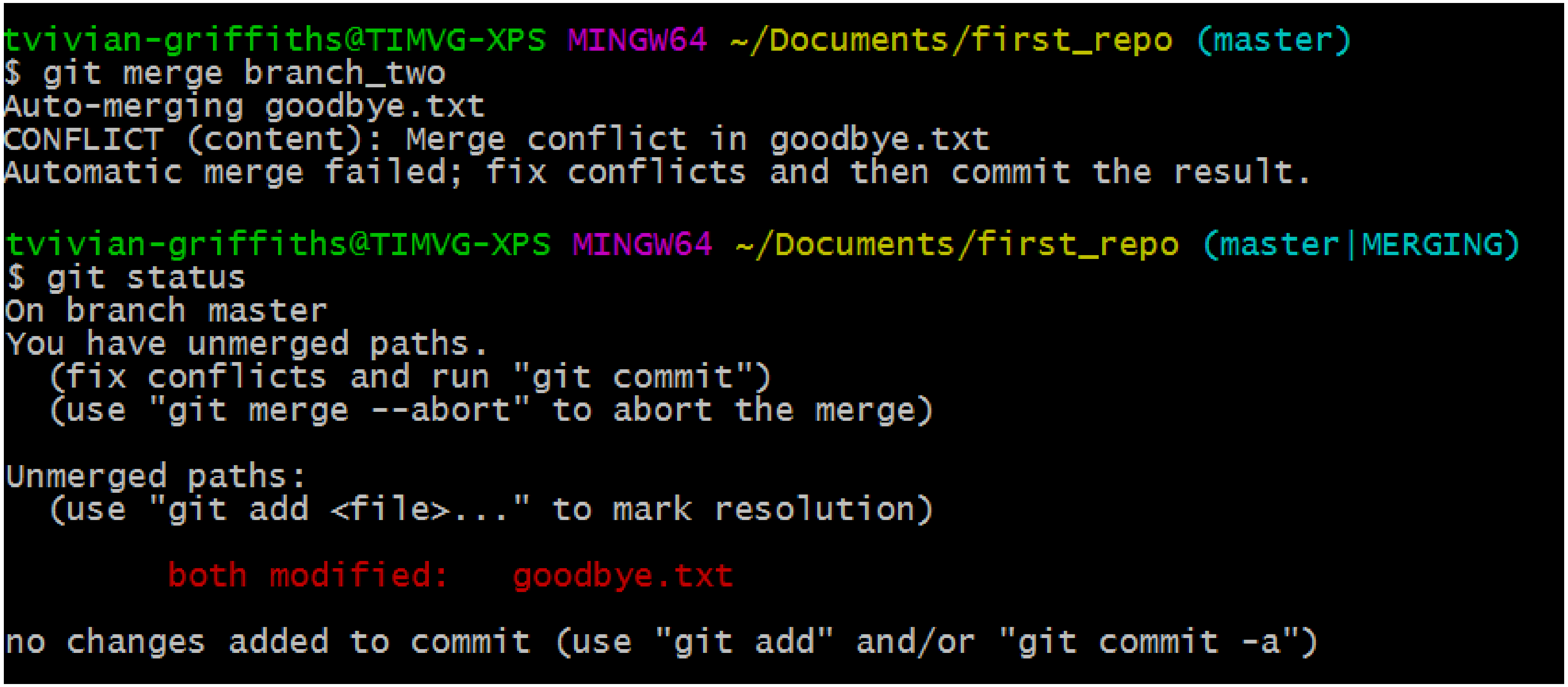
- Last time the branches diverged by editing different files
- **But** if we change the **same lines** in the **same file**, we get a **merge conflict**
- Git cannot decide which one to keep – we need to fix this manually

Preparing the conflict

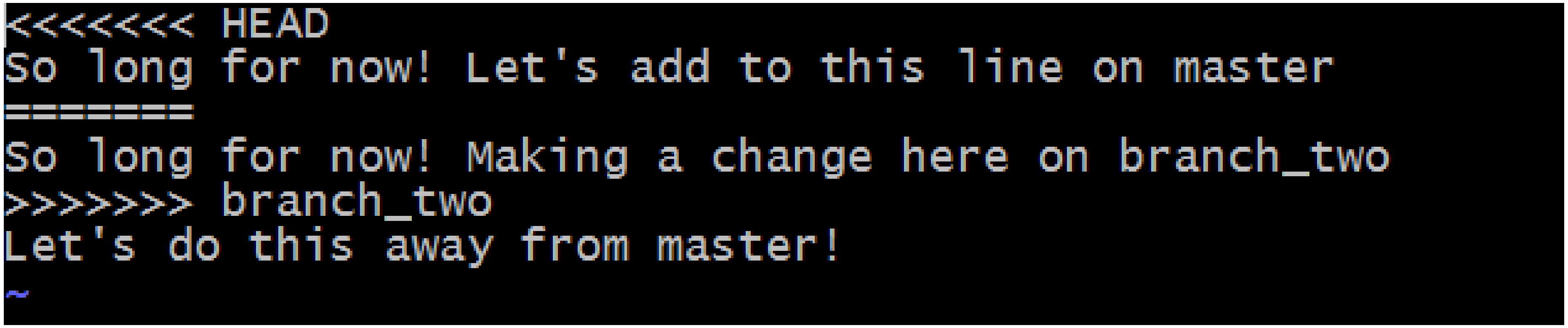


```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git hist
* 718c2fe (branch_two) Changing first line of goodbye.txt on branch_two
| * e5a4f8f (HEAD -> master) Changing first line of goodbye.txt on master
|/
*   a902edd Merge message - typed in VIM
|\
| * 41f9630 (branch_one) Changing goodbye.txt on branch_one
* | 9575133 Making a change on master to hello.txt
|/
* 4e85b56 Adding hello2 and goodbye changes
* e057879 Adding the hello file changes
* e3dd088 Adding the file 'hello.txt' to the repo
```

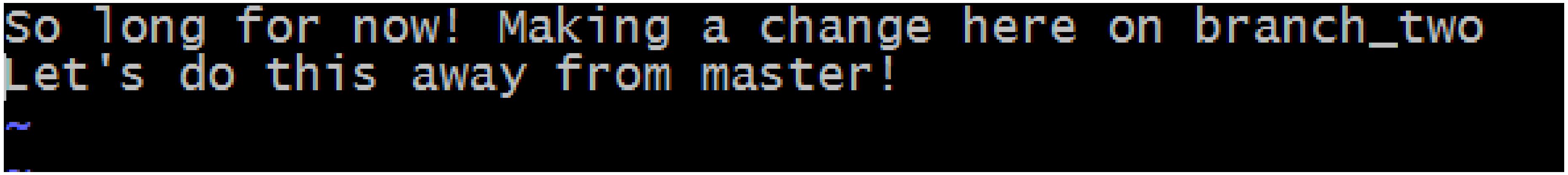
Trying to merge


```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git merge branch_two
Auto-merging goodbye.txt
CONFLICT (content): Merge conflict in goodbye.txt
Automatic merge failed; fix conflicts and then commit the result.

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master|MERGING)
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   goodbye.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Fix in VIM – keeping **branch**



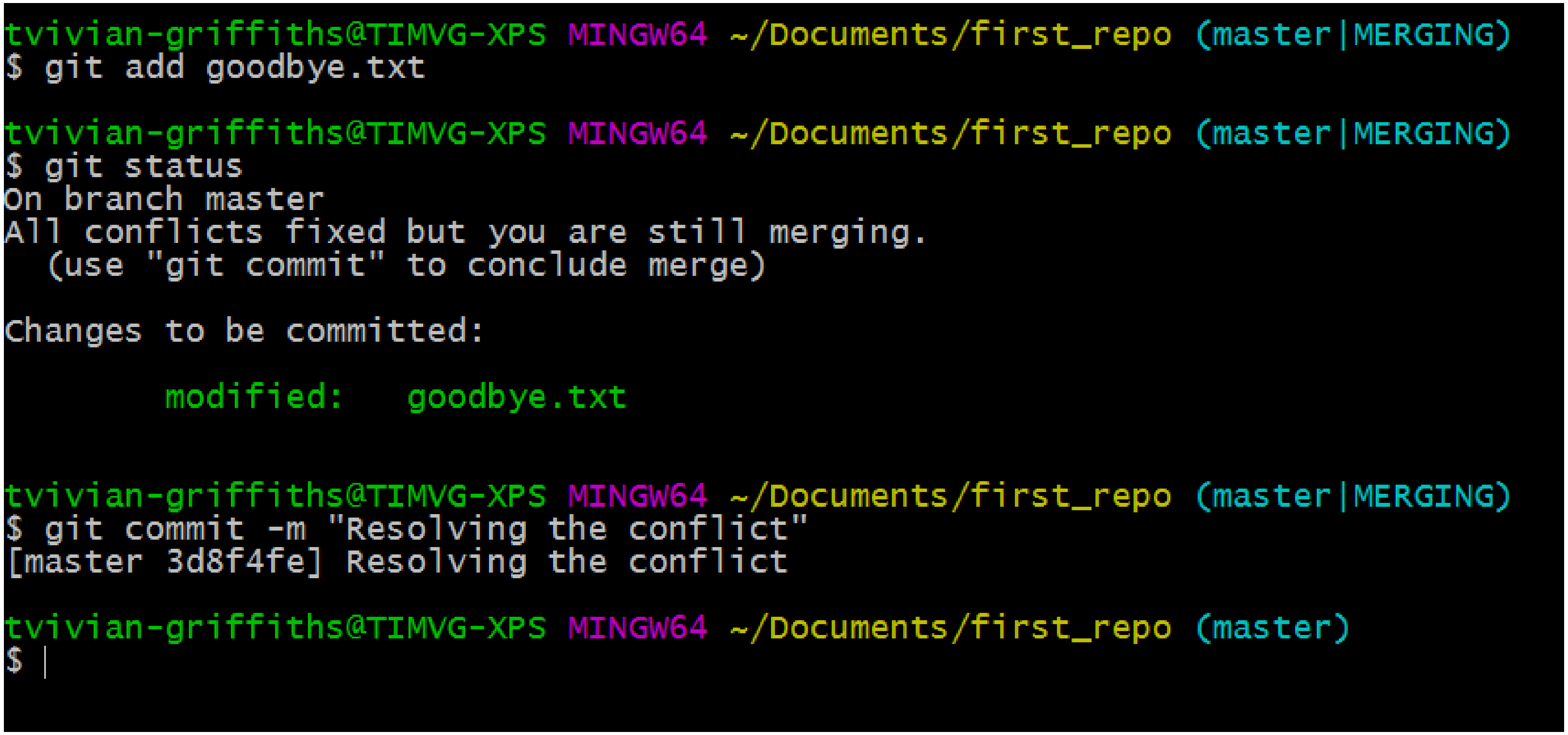
```
<<<<<<< HEAD
So long for now! Let's add to this line on master
=======
So long for now! Making a change here on branch_two
>>>>>>> branch_two
Let's do this away from master!
~
```



```
So long for now! Making a change here on branch_two
Let's do this away from master!
~
```

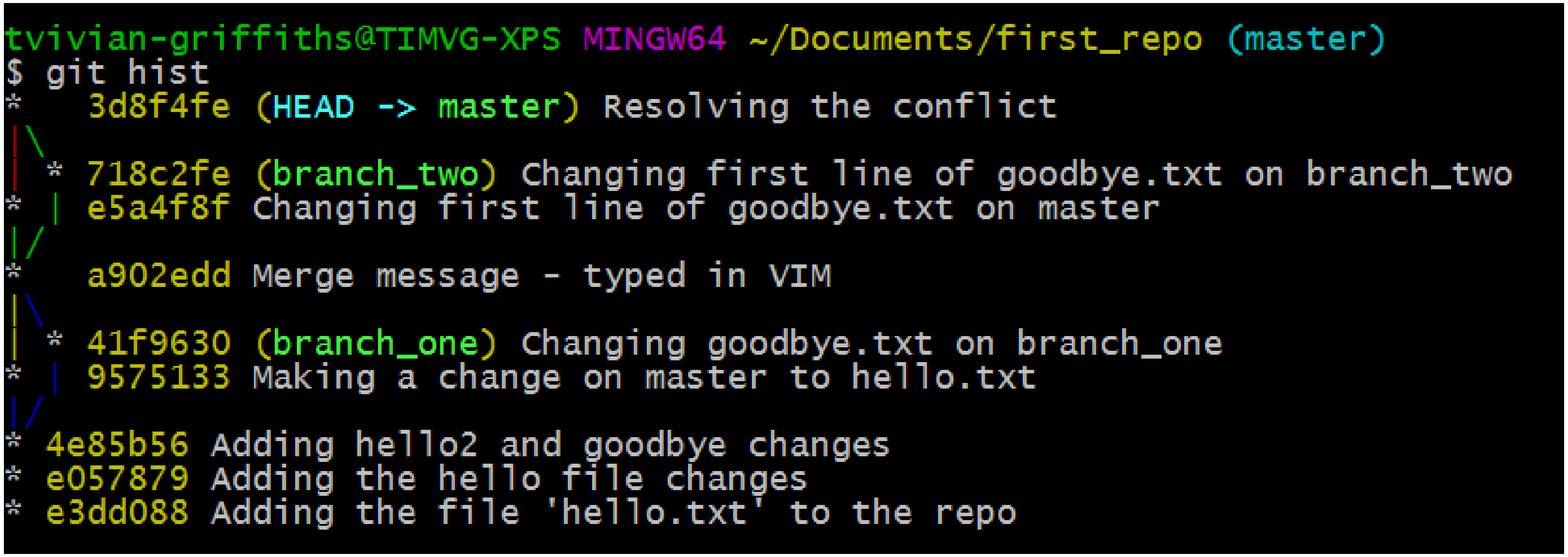
- Tip – use dd to delete a whole line in VIM

Commit the resolution



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master|MERGING)
$ git add goodbye.txt

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master|MERGING)
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

        modified:   goodbye.txt


tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master|MERGING)
$ git commit -m "Resolving the conflict"
[master 3d8f4fe] Resolving the conflict

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ 
```

Viewing the resolved merge



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git hist
*   3d8f4fe (HEAD -> master) Resolving the conflict
|\
| * 718c2fe (branch_two) Changing first line of goodbye.txt on branch_two
* | e5a4f8f Changing first line of goodbye.txt on master
|/
*   a902edd Merge message - typed in VIM
|\
| * 41f9630 (branch_one) Changing goodbye.txt on branch_one
* | 9575133 Making a change on master to hello.txt
|/
* 4e85b56 Adding hello2 and goodbye changes
* e057879 Adding the hello file changes
* e3dd088 Adding the file 'hello.txt' to the repo
```

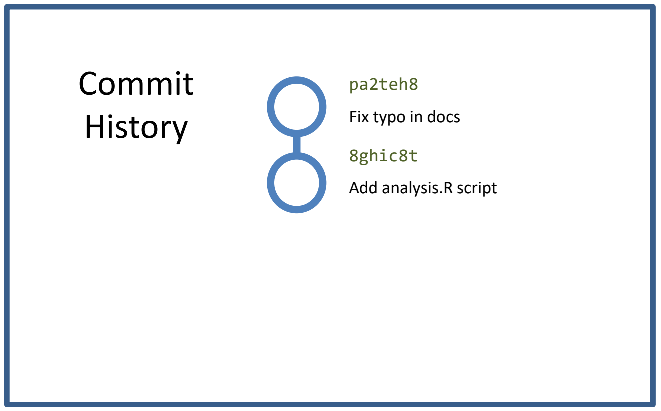
VOL+
STOP/ MODE
PREV
VOL-
SETUP
ENTER/ SAVE
CH+
CH-
NEXT
3
2
6
0 10+
1
5
9
4
8
7

Remote Git

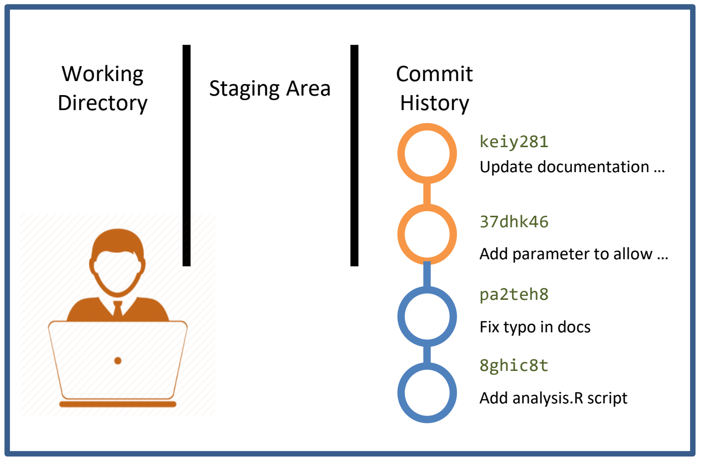
Distributed Version Control

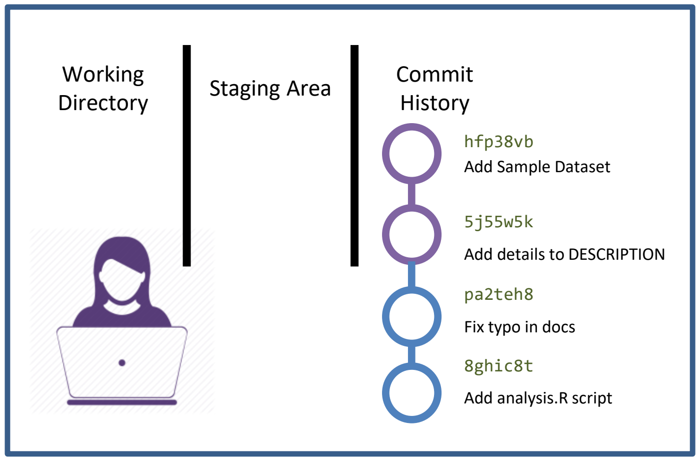
- Git repos are self contained entities, where each can have there own separate history.
- Each user has a repo locally, and a common setup is to have another nominated to be a “central repository”.
- References to other repositories are called “remotes"
- Users can push and pull code to/from these remote repositories.

Distributed Version Control



Central Repo is a remote named "origin" (Note No Working Directory)





GitHub

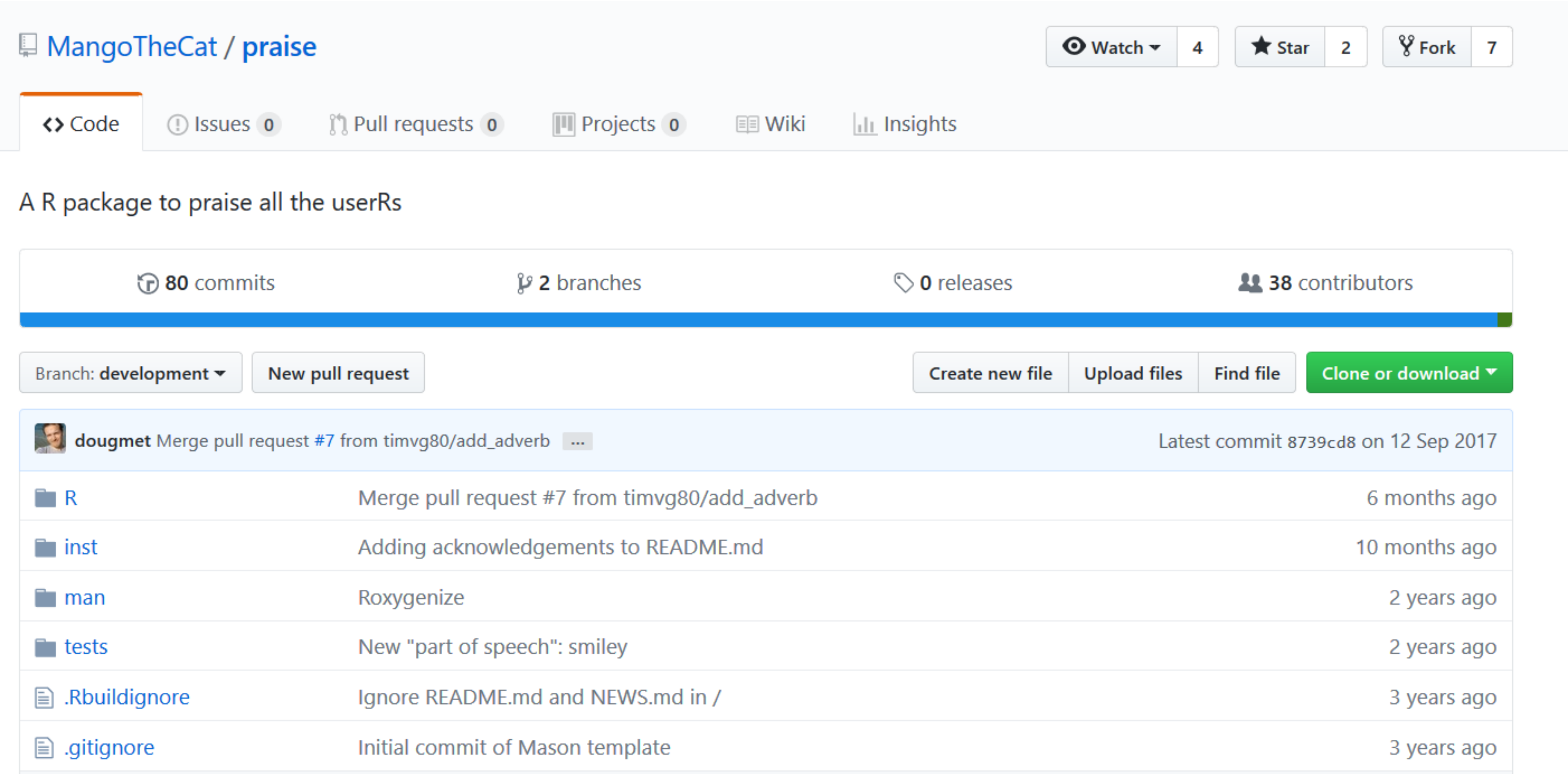
- Web-based hosting platform for projects
  - Phenomenal for collaboration
  - Used for many open-source projects
    - The TidyVerse packages are on GitHub
  - Free for personal usage with open repositories

- Other options: GitLab, BitBucket, …

Not just code

- git & GitHub are great for R Markdown
- Data analysis, presentations, etc.
- Scientific articles (not in word)
- Anything that is text, really: see e.g. Awesome R

Files



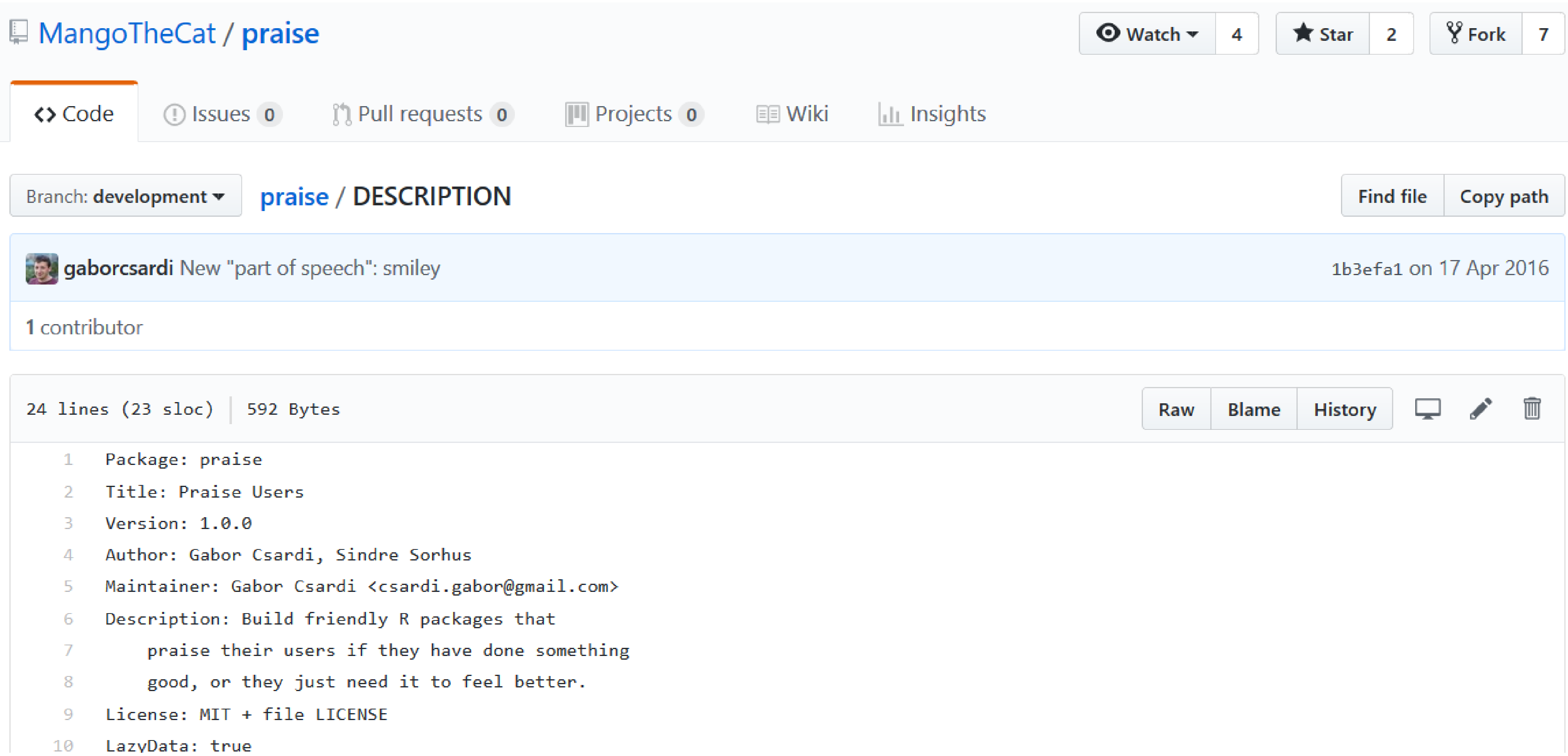
MangoTheCat / praise

A R package to praise all the userRs

dougmet Merge pull request #7 from timvg80/add_adverb ... Latest commit 8739cd8 on 12 Sep 2017

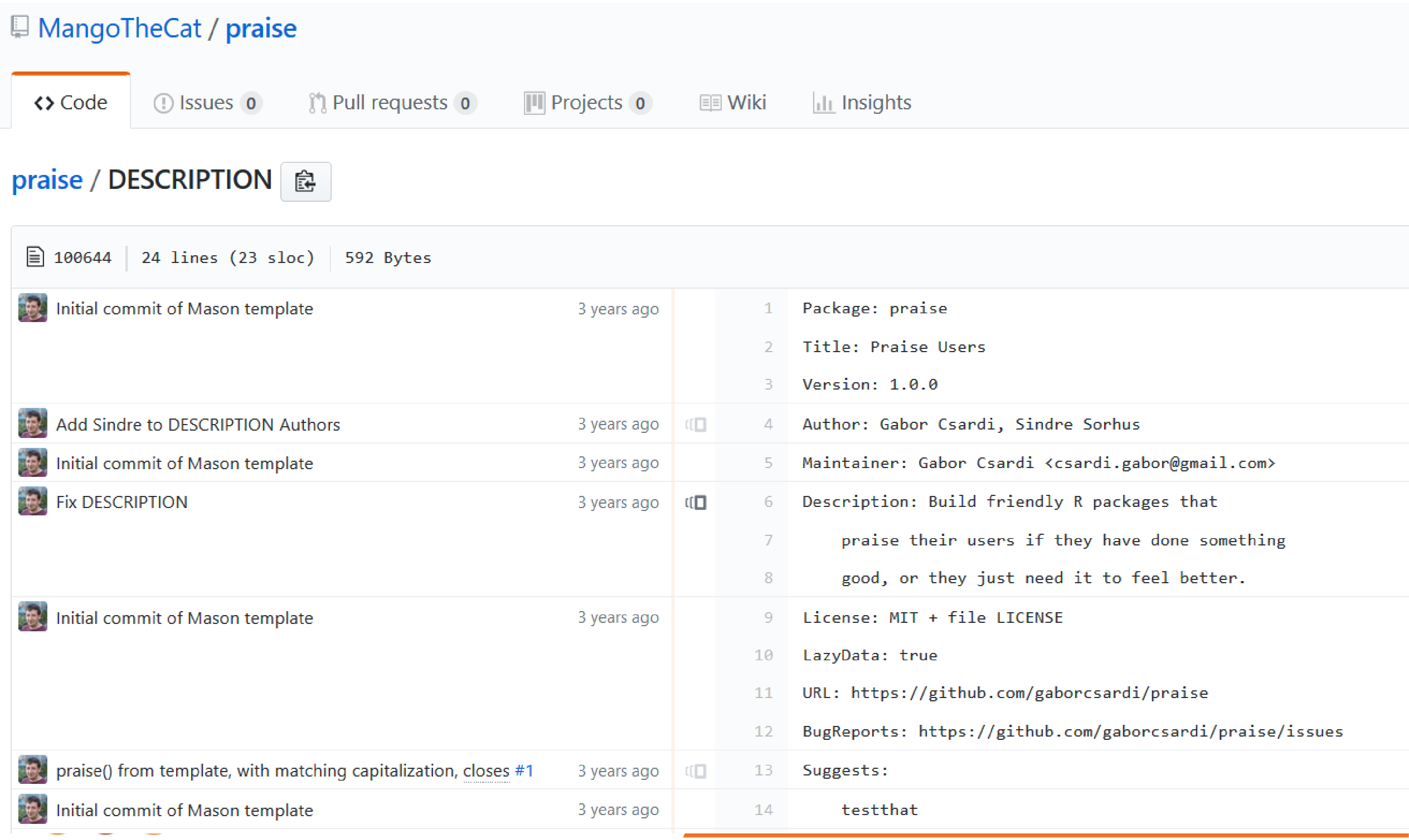
R	Merge pull request #7 from timvg80/add_adverb	6 months ago
inst	Adding acknowledgements to README.md	10 months ago
man	Roxygenize	2 years ago
tests	New "part of speech": smiley	2 years ago
.Rbuildignore	Ignore README.md and NEWS.md in /	3 years ago
.gitignore	Initial commit of Mason template	3 years ago

Files



```
Package: praise
Title: Praise Users
Version: 1.0.0
Author: Gabor Csardi, Sindre Sorhus
Maintainer: Gabor Csardi <csardi.gabor@gmail.com>
Description: Build friendly R packages that
    praise their users if they have done something
    good, or they just need it to feel better.
License: MIT + file LICENSE
LazyData: true
```

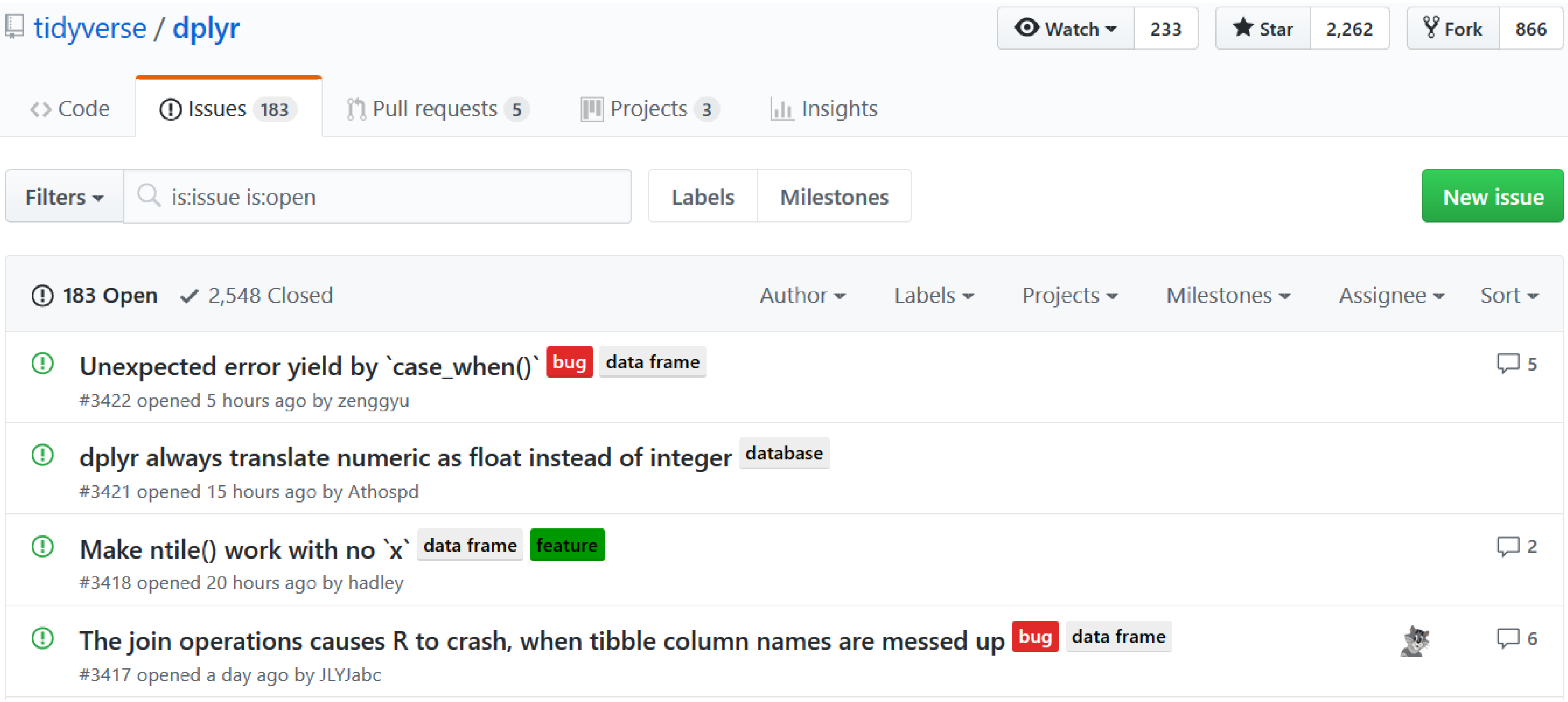
History



100644 | 24 lines (23 sloc) | 592 Bytes

Initial commit of Mason template	3 years ago	Package: praise Title: Praise Users Version: 1.0.0
Add Sindre to DESCRIPTION Authors	3 years ago	Author: Gabor Csardi, Sindre Sorhus
Initial commit of Mason template	3 years ago	Maintainer: Gabor Csardi <csardi.gabor@gmail.com>
Fix DESCRIPTION	3 years ago	Description: Build friendly R packages that praise their users if they have done something good, or they just need it to feel better.
Initial commit of Mason template	3 years ago	License: MIT + file LICENSE LazyData: true URL: https://github.com/gaborcsardi/praise BugReports: https://github.com/gaborcsardi/praise/issues
praise() from template, with matching capitalization, closes #1	3 years ago	Suggests:
Initial commit of Mason template	3 years ago	testthat

Issues


tidyverse / dplyr

Watch 233 | Star 2,262 | Fork 866

Code | Issues 183 | Pull requests 5 | Projects 3 | Insights

Filters | is:issue is:open | Labels | Milestones | New issue

183 Open ✓ 2,548 Closed | Author | Labels | Projects | Milestones | Assignee | Sort

Unexpected error yield by `case_when()` bug data frame — 5
#3422 opened 5 hours ago by zenggyu

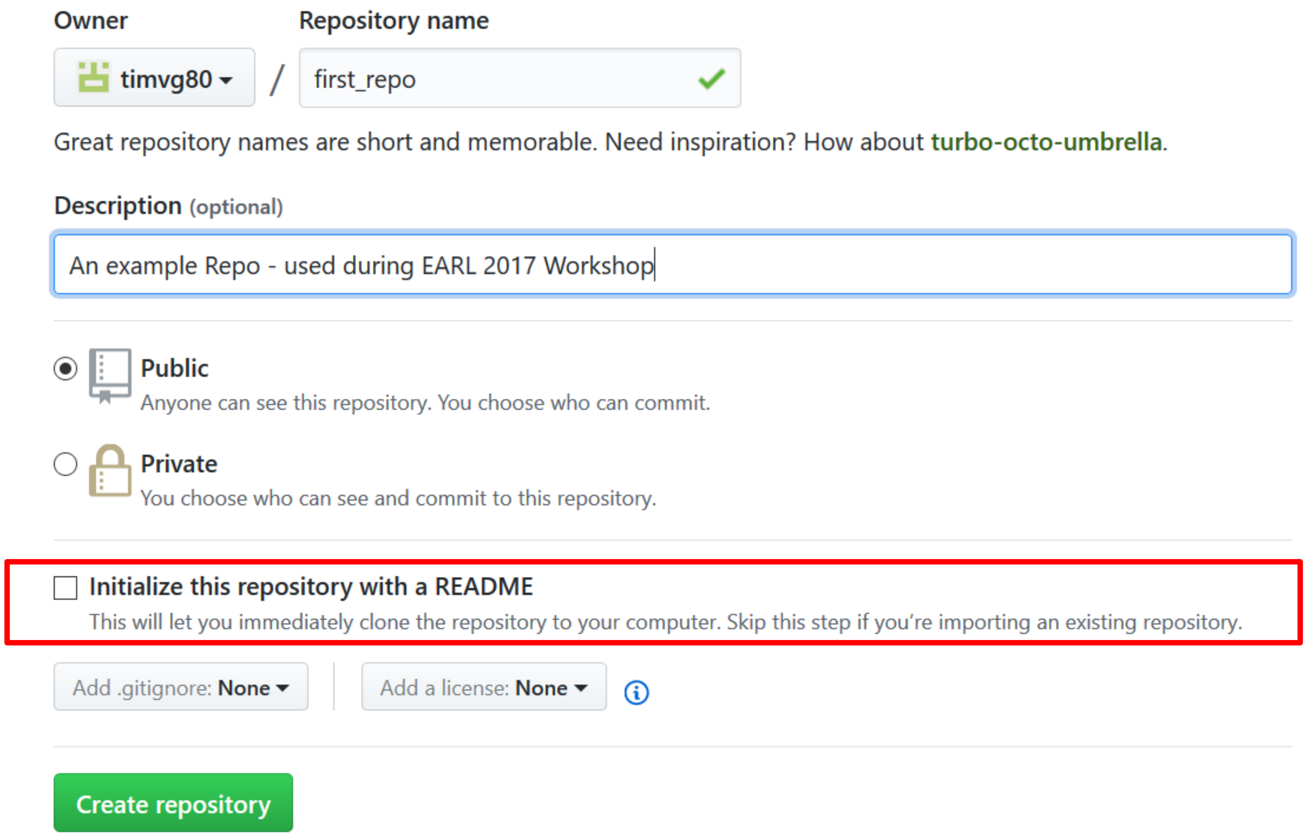
dplyr always translate numeric as float instead of integer database
#3421 opened 15 hours ago by Athospd

Make ntile() work with no `x` data frame feature — 2
#3418 opened 20 hours ago by hadley

The join operations causes R to crash, when tibble column names are messed up bug data frame — 6
#3417 opened a day ago by JLYJabc

Start a new Repository in GitHub

- **Profile -> Repositories -> New**



Owner

timvg80 ▾ / first_repo ✓

Repository name

Great repository names are short and memorable. Need inspiration? How about turbo-octo-umbrella.

Description (optional)

An example Repo - used during EARL 2017 Workshop

● 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: None ▾

DON'T!!!!

Cloning

- Copy a remote repository

```
$ git clone
https://github.com/adfi/dummy_repo.git
```

- Creates local repository in `dummy_repo`
- cd into this directory then:

```
  $ git remote -v
```

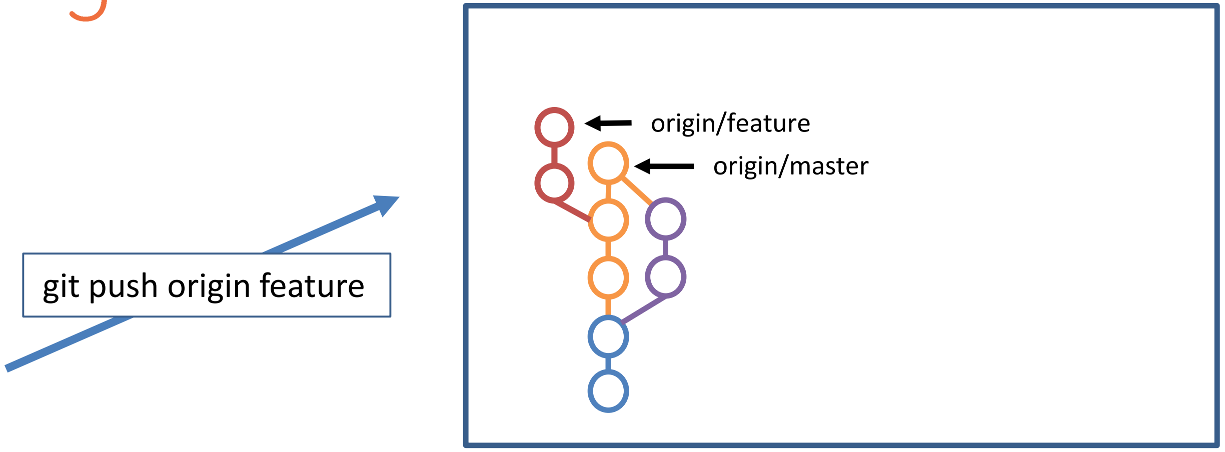
Exercise

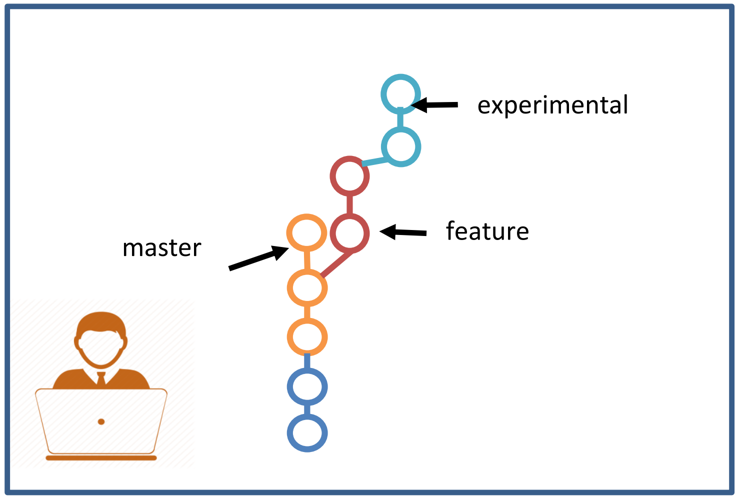
- Log into GitHub
- Create a new repository
- Add a README.md file
- Add some text and commit
- Clone this repository to your computer

Communicating from local to remote

- `$ git push`
  - Send info from local to remote
- `$ git fetch`
  - Get the new information from the remote
- `$ git merge`
  - Merge the remote's information into local
- `$ git pull`
  - `fetch` and `merge` combined (be careful!!!)

Branching



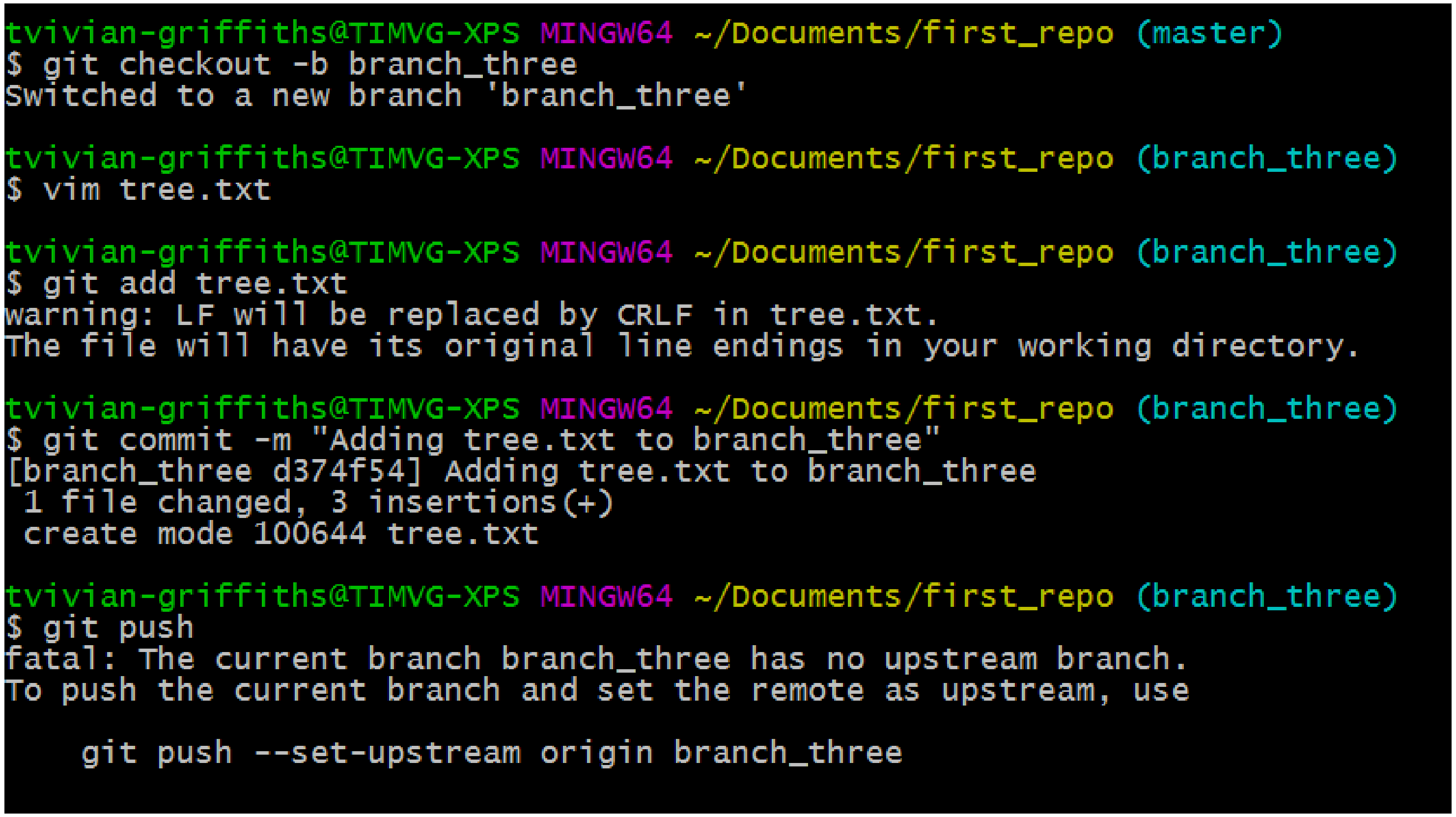


- branches can be pushed/pulled individually.
- branches on different remotes are completely independent
- Must set local branches to “track” remote branches.

```
$ git push --set-upstream origin
  <branchname>
```

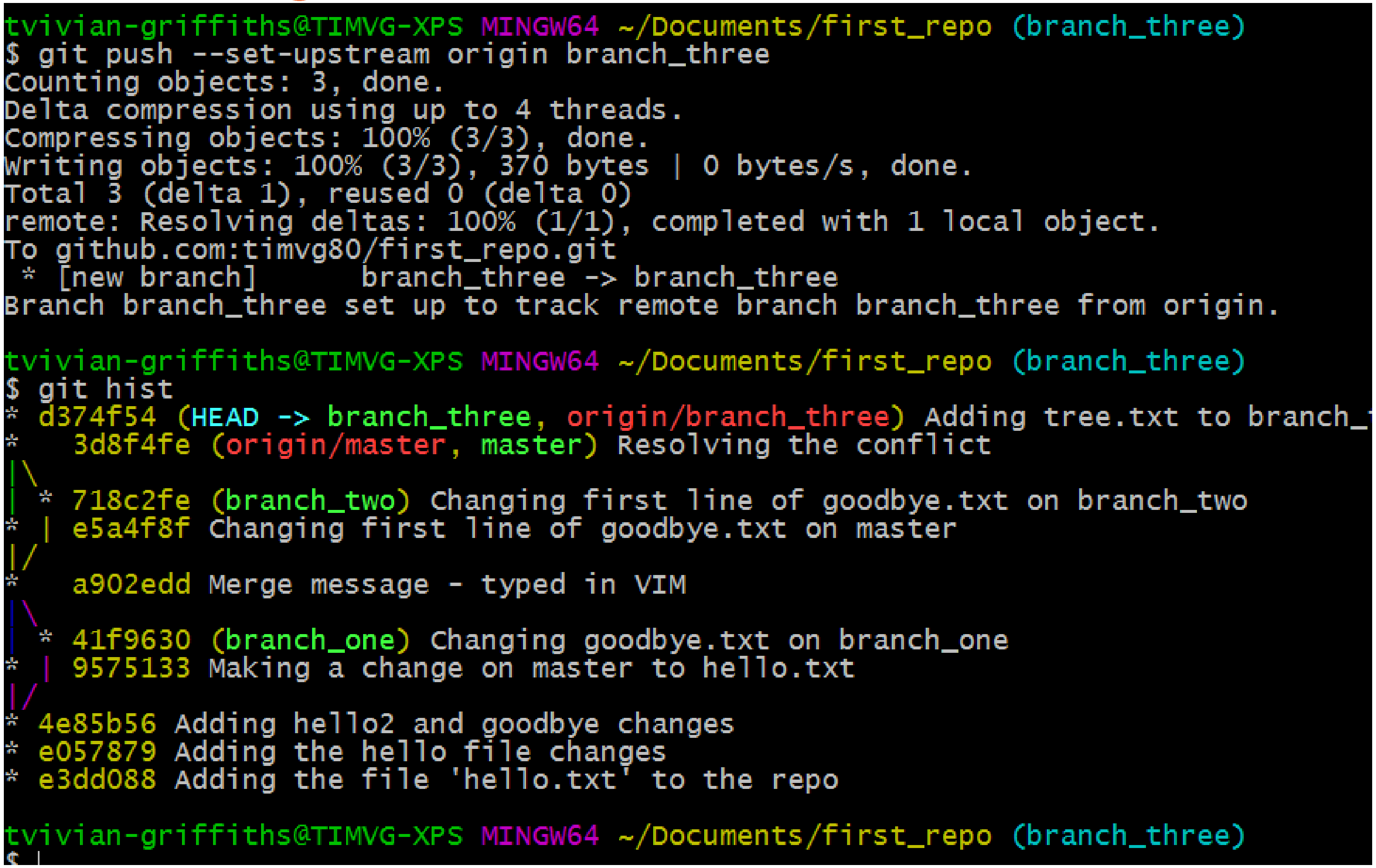
Pushing a new branch to remote

- Git continues to help you!



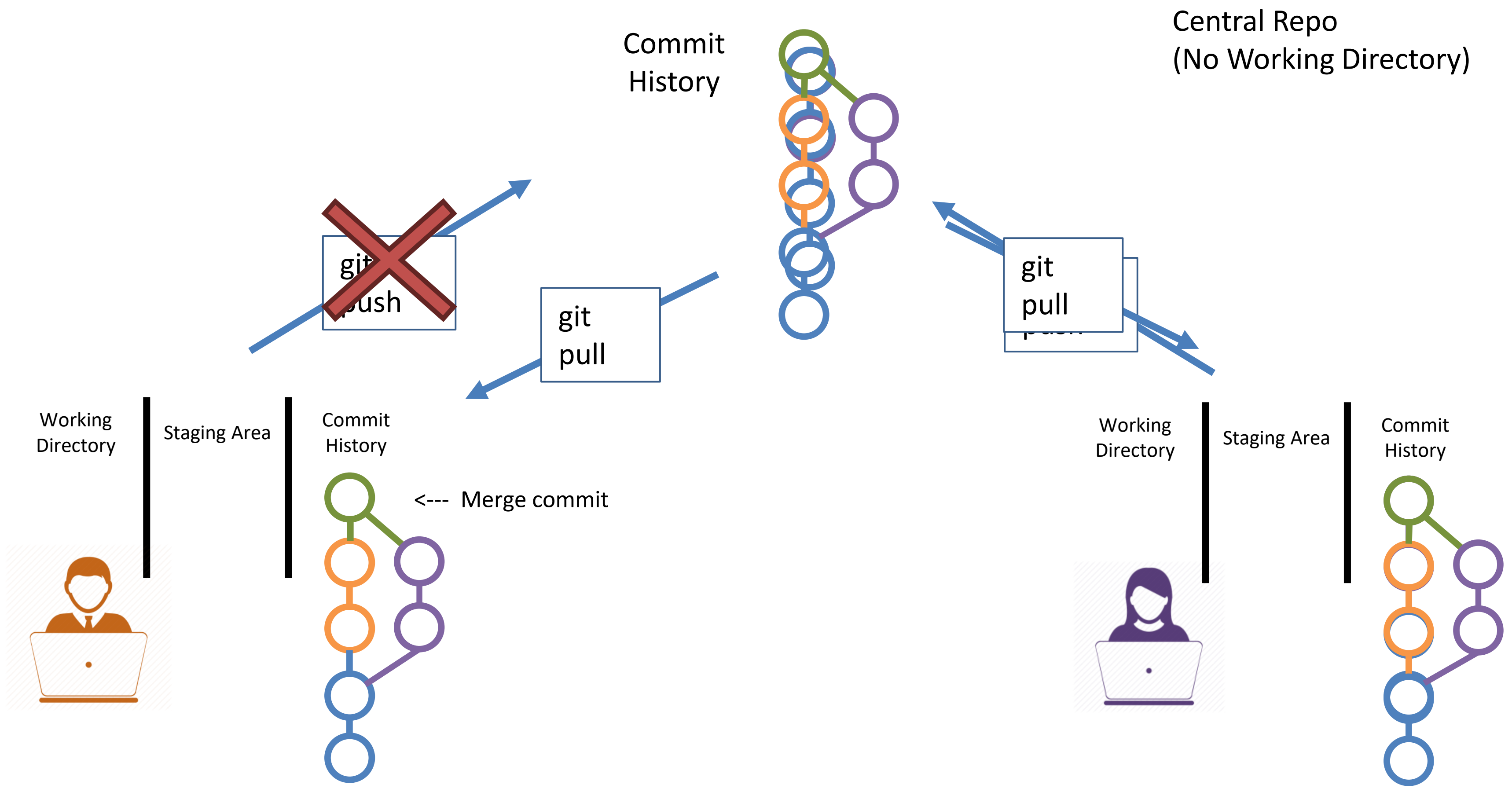
```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (master)
$ git checkout -b branch_three
Switched to a new branch 'branch_three'

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_three)
$ vim tree.txt

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_three)
$ git add tree.txt
warning: LF will be replaced by CRLF in tree.txt.
The file will have its original line endings in your working directory.

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_three)
$ git commit -m "Adding tree.txt to branch_three"
[branch_three d374f54] Adding tree.txt to branch_three
 1 file changed, 3 insertions(+)
 create mode 100644 tree.txt

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_three)
$ git push
fatal: The current branch branch_three has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin branch_three
```

Pushing success



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_three)
$ git push --set-upstream origin branch_three
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 370 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:timvg80/first_repo.git
 * [new branch]      branch_three -> branch_three
Branch branch_three set up to track remote branch branch_three from origin.

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_three)
$ git hist
* d374f54 (HEAD -> branch_three, origin/branch_three) Adding tree.txt to branch_
*   3d8f4fe (origin/master, master) Resolving the conflict
|\
| * 718c2fe (branch_two) Changing first line of goodbye.txt on branch_two
* | e5a4f8f Changing first line of goodbye.txt on master
|/
*   a902edd Merge message - typed in VIM
|\
| * 41f9630 (branch_one) Changing goodbye.txt on branch_one
* | 9575133 Making a change on master to hello.txt
|/
* 4e85b56 Adding hello2 and goodbye changes
* e057879 Adding the hello file changes
* e3dd088 Adding the file 'hello.txt' to the repo

tvivian-griffiths@TIMVG-XPS MINGW64 ~/Documents/first_repo (branch_three)
$ |
```

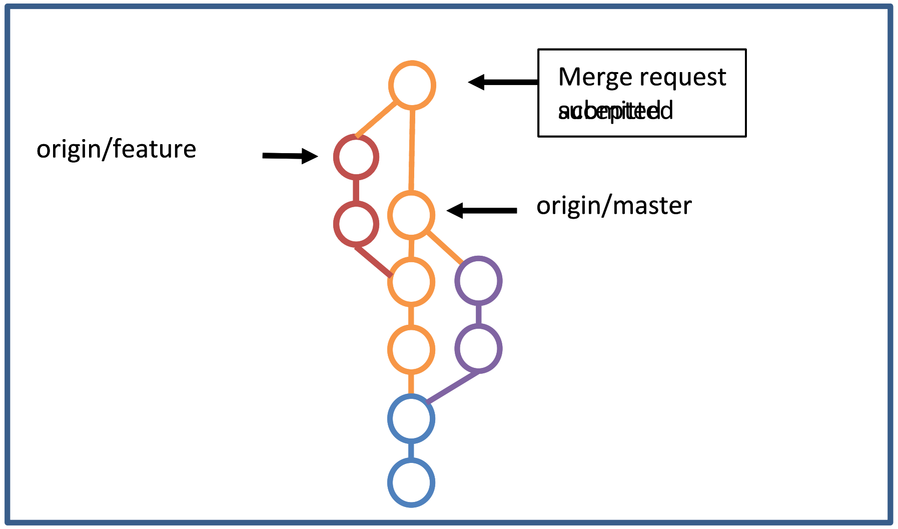
Merges with remotes



Collaboration + Git = Merge Requests

Merge requests

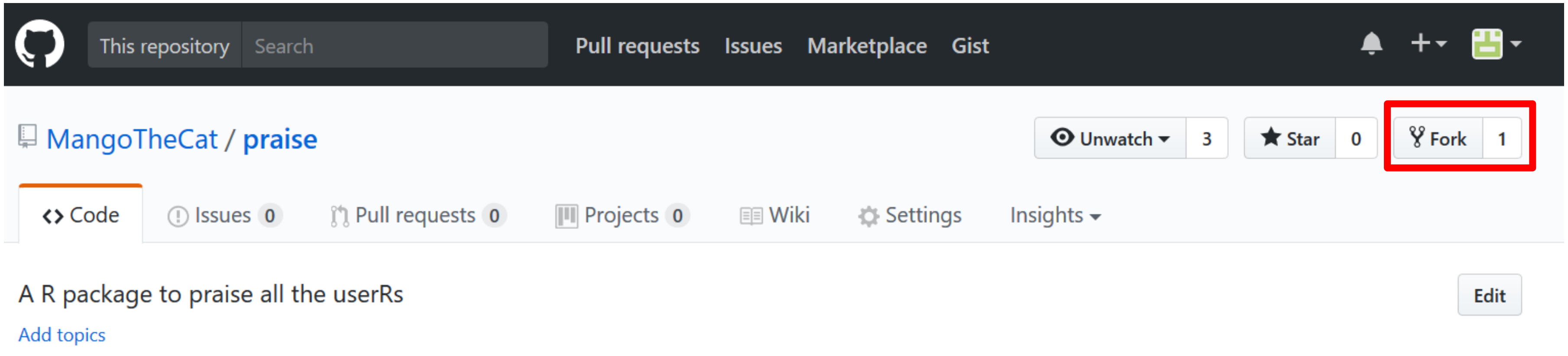
- Common development practice
- QC check before merging into master branch on the origin repo
- GitHub provides tools to review changes and accept merge.
- Often requires right "role permissions"



Forking an existing repo

- Method when contributing to existing project
  - You don't make changes to that repo
  - Instead – you bring it into your GitHub account
  - This process is called **forking**
- Search for the **praise** package on the **MangoTheCat** account

Forking in the GitHub UI



A R package to praise all the userRs

- Do this when already signed into your account

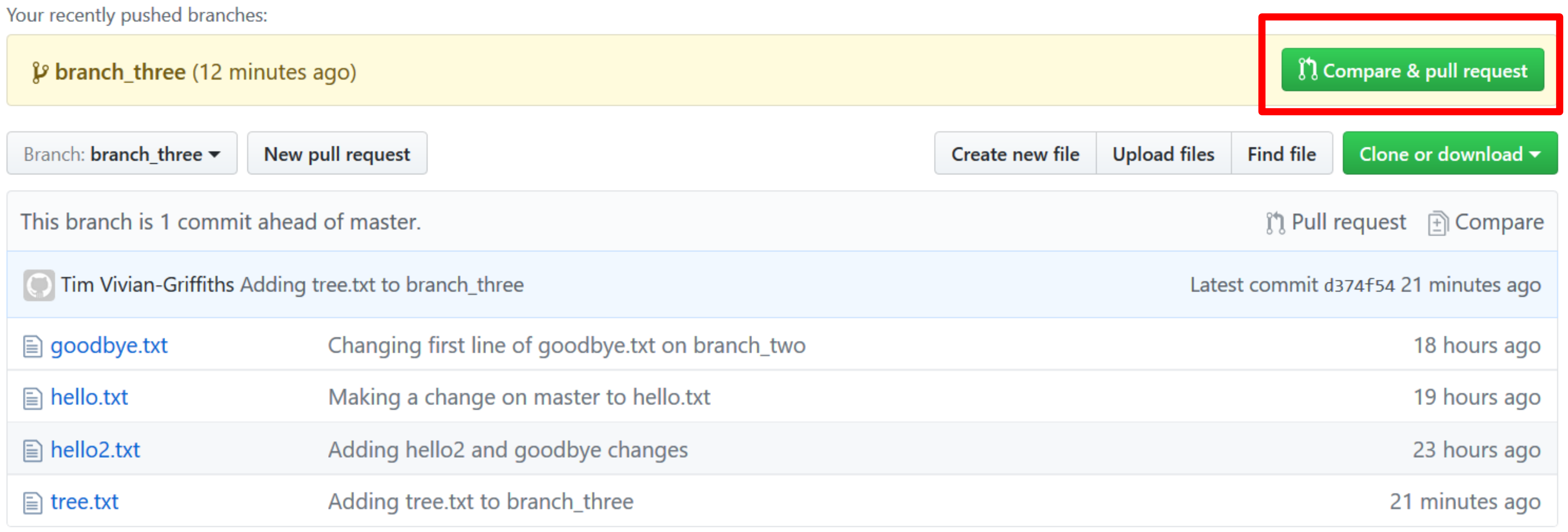
Making a contribution

- The praise package allows for very simple changes to be made
- We can just add individual words to existing lists
- We will do this and then push to our forked repo
- Remember to use branches!!!

Exercise

- Clone the forked praise repository to your laptop
- Create a branch
- Add a word to either *adjective.R* , *adverb.R* or *exclamation.R*
- Add, commit and push your changes

Making pull request



Your recently pushed branches:

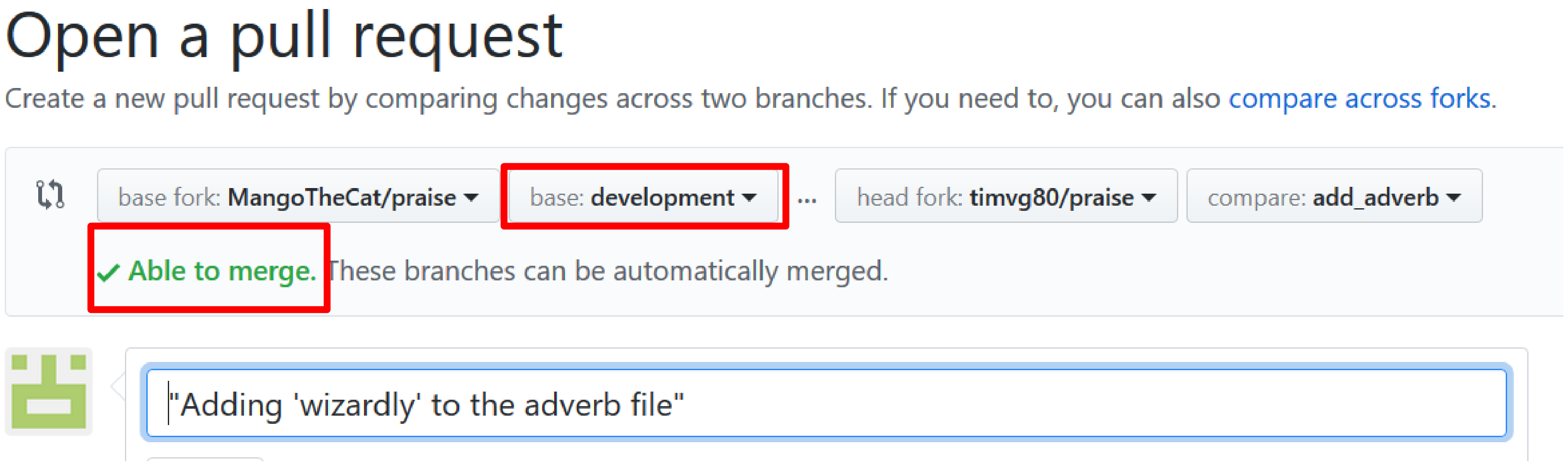
branch_three (12 minutes ago)

This branch is 1 commit ahead of master.

Tim Vivian-Griffiths Adding tree.txt to branch_three — Latest commit d374f54 21 minutes ago

goodbye.txt	Changing first line of goodbye.txt on branch_two	18 hours ago
hello.txt	Making a change on master to hello.txt	19 hours ago
hello2.txt	Adding hello2 and goodbye changes	23 hours ago
tree.txt	Adding tree.txt to branch_three	21 minutes ago

GitHub UI for pull request



Open a pull request

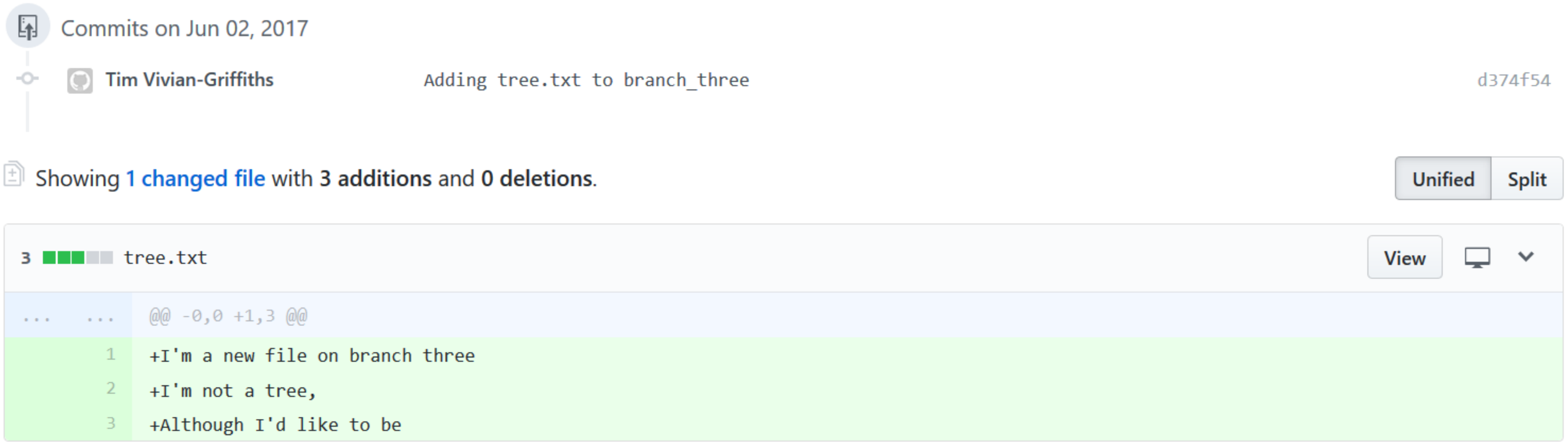
Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base fork: **MangoTheCat/praise** ▾ base: **development** ▾ ... head fork: **timvg80/praise** ▾ compare: **add_adverb** ▾

✓ Able to merge. These branches can be automatically merged.

"Adding 'wizardly' to the adverb file"

View the changes



Commits on Jun 02, 2017

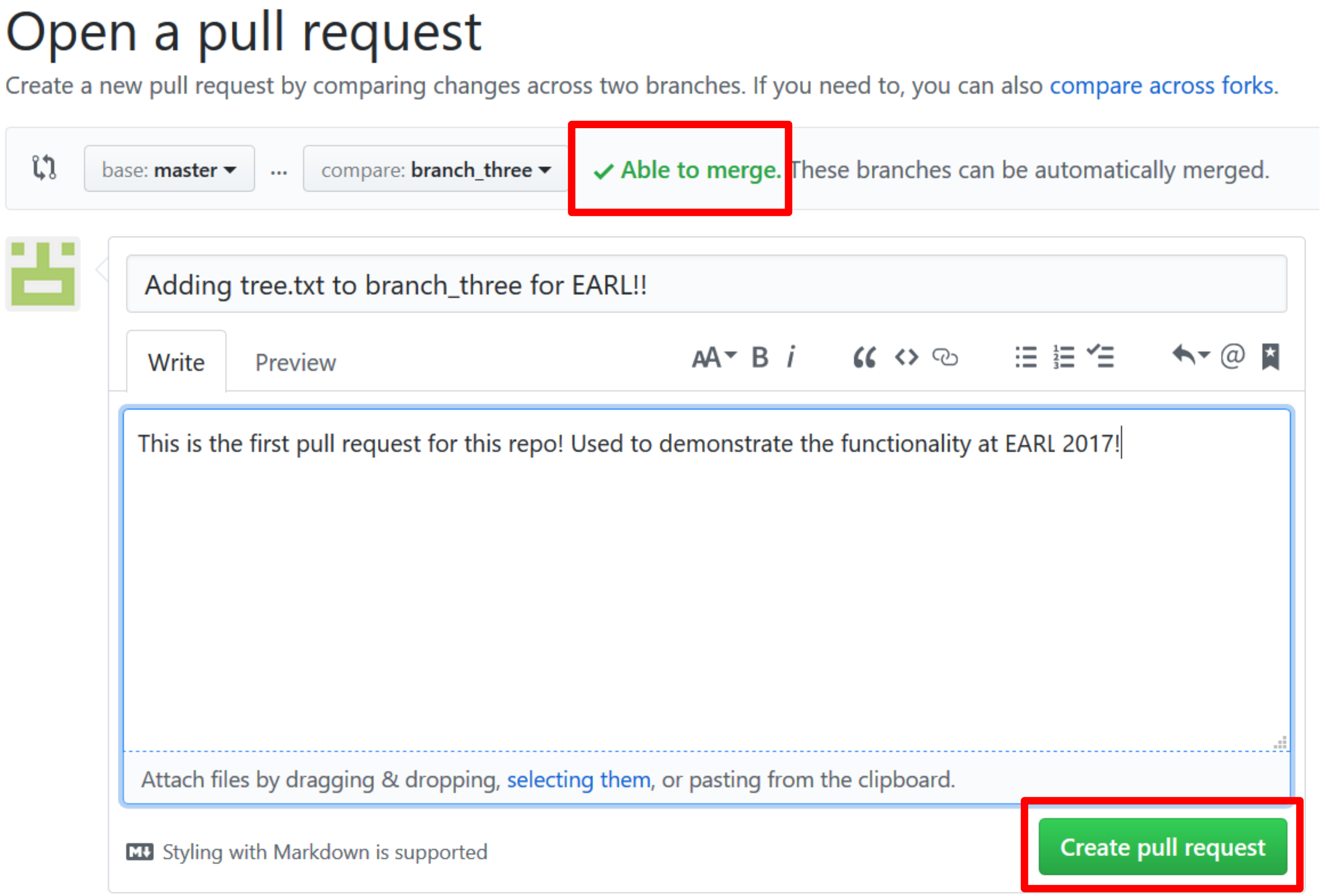
Tim Vivian-Griffiths   `Adding tree.txt to branch_three`   d374f54

Showing 1 changed file with 3 additions and 0 deletions.

3 tree.txt

```
@@ -0,0 +1,3 @@
+I'm a new file on branch three
+I'm not a tree,
+Although I'd like to be
```

Check and request!



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: **master** ... compare: **branch_three** ✓ Able to merge. These branches can be automatically merged.

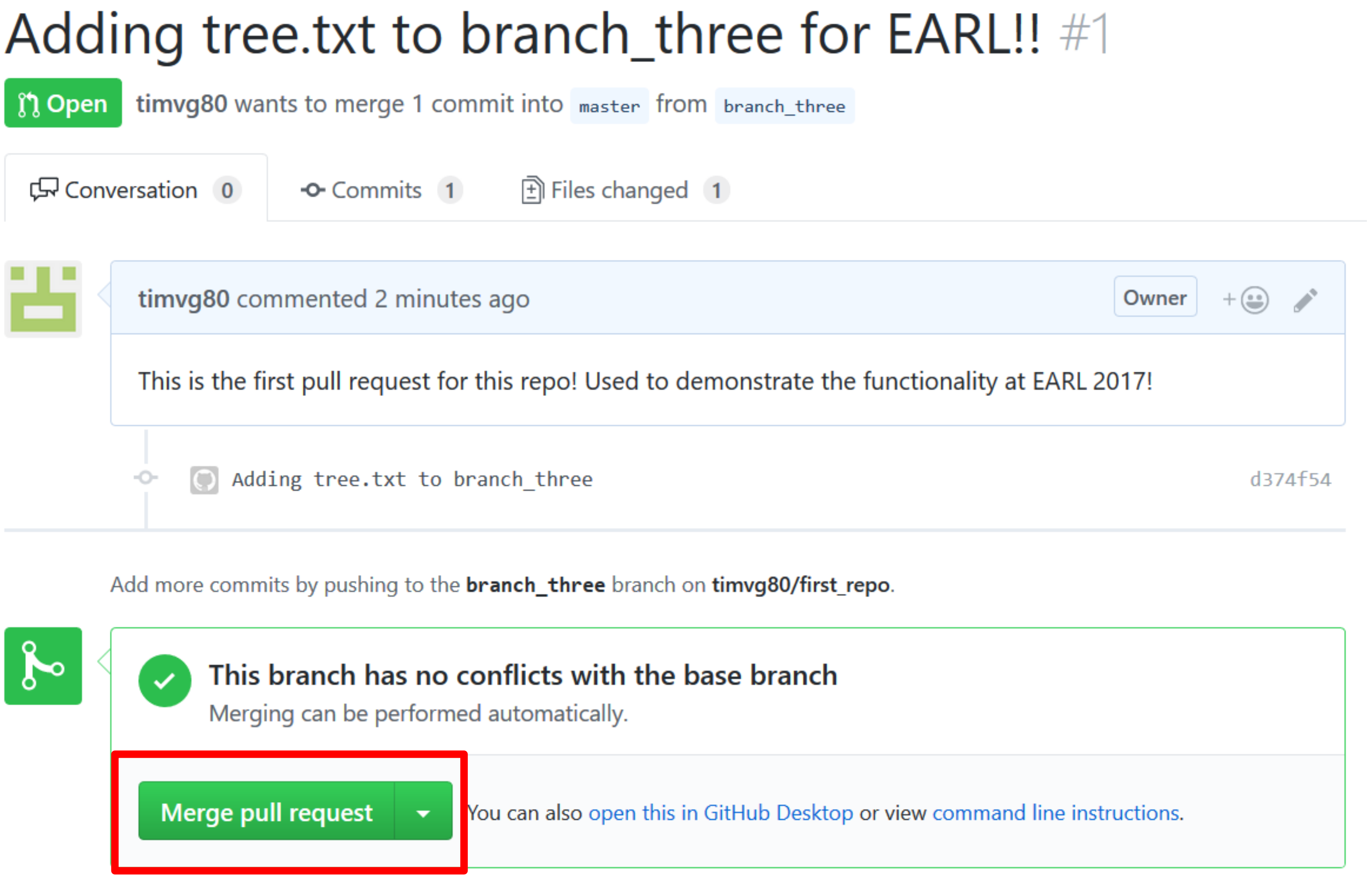
Adding tree.txt to branch_three for EARL!!

This is the first pull request for this repo! Used to demonstrate the functionality at EARL 2017!

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Accepting the request: A colleague should really do this



Adding tree.txt to branch_three for EARL!! #1

Open timvg80 wants to merge 1 commit into `master` from `branch_three`

timvg80 commented 2 minutes ago

This is the first pull request for this repo! Used to demonstrate the functionality at EARL 2017!

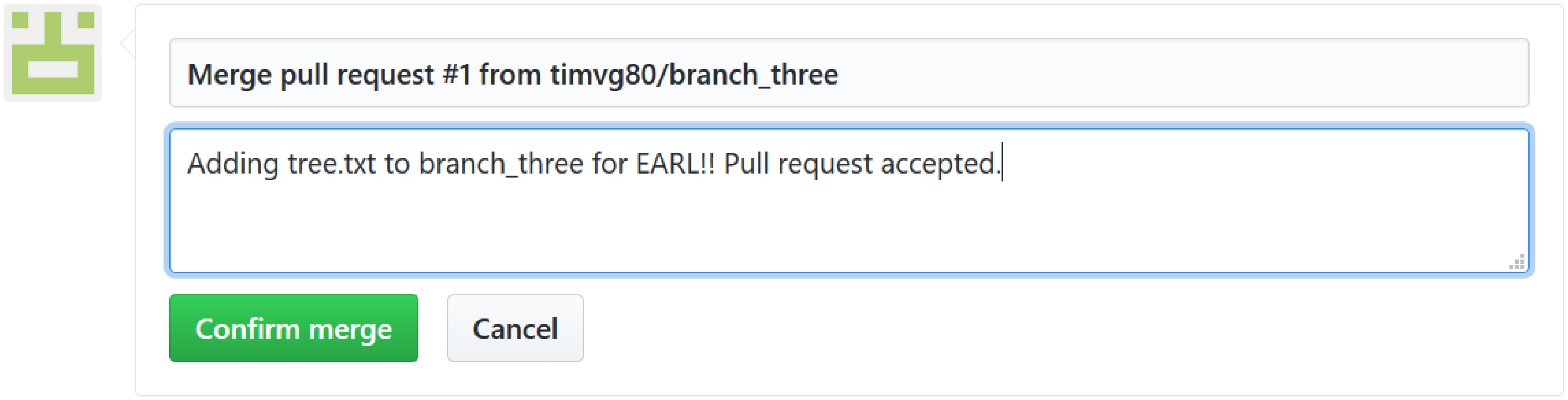
`Adding tree.txt to branch_three` d374f54

Add more commits by pushing to the **branch_three** branch on **timvg80/first_repo**.

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

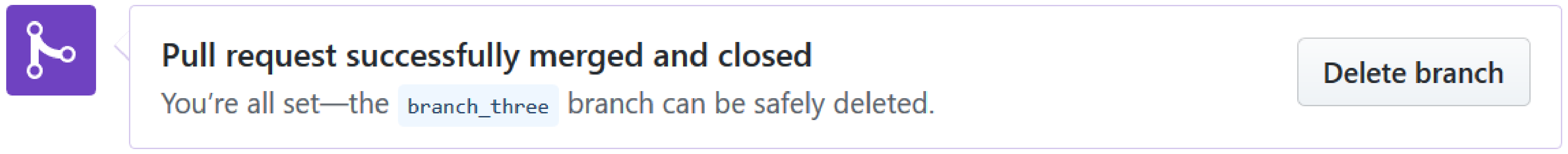
Accepting the request cont.



Merge pull request #1 from timvg80/branch_three

Adding tree.txt to branch_three for EARL!! Pull request accepted.

Cleaning up… delete the branch



Pull request successfully merged and closed

You're all set—the `branch_three` branch can be safely deleted.

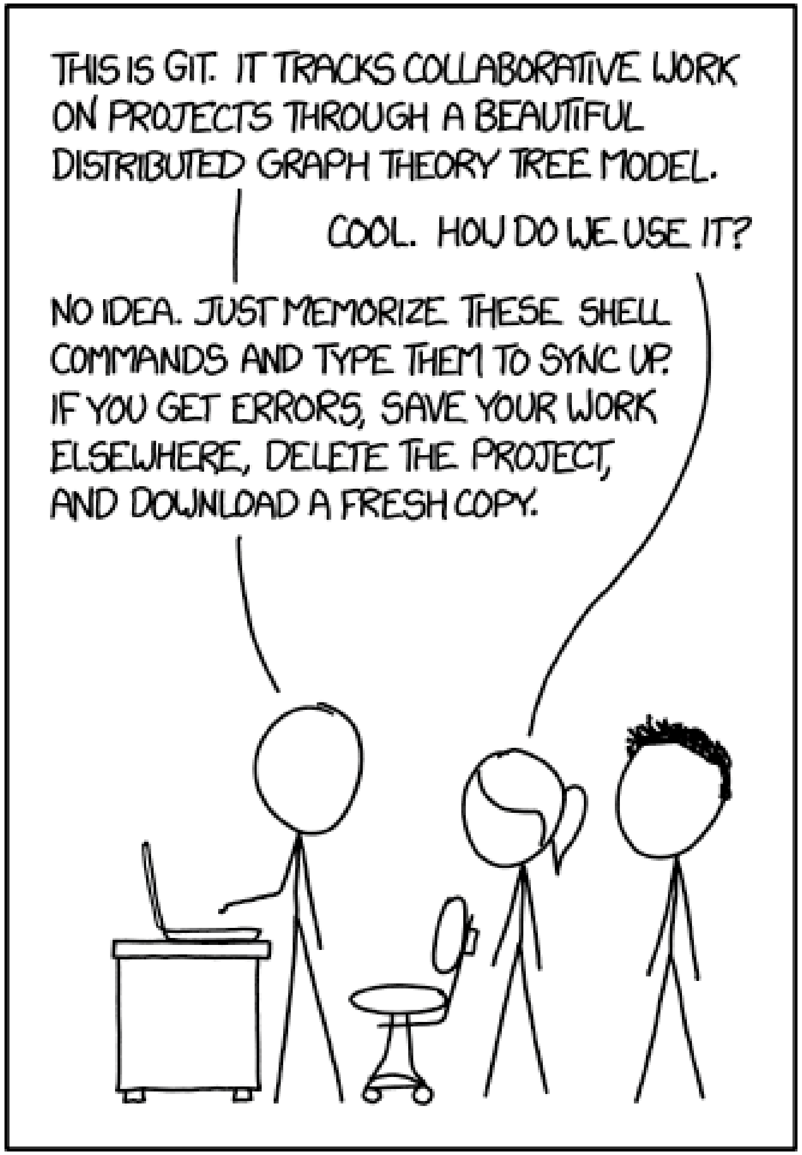
- Now we need to update our local repo

Good Practices

- Work on a branch whenever possible
  - If your about to try implementing a new feature/function or bug fix
  - branches should be **short lived**
- Master should represent the most "production ready" code
- Commit early and often with good commit messages
- Review each others work through merge requests

Good Practices

- Some actions in Git can seem scary
  - Make a back-up branch if you are unsure
  - You can always checkout back to this branch
- When working on an Open Source project:
  - Read their contribution guidelines
  - Learning about tracking **issues**
  - Adhere to their standards
- Good luck!


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.

https://xkcd.com/1597/

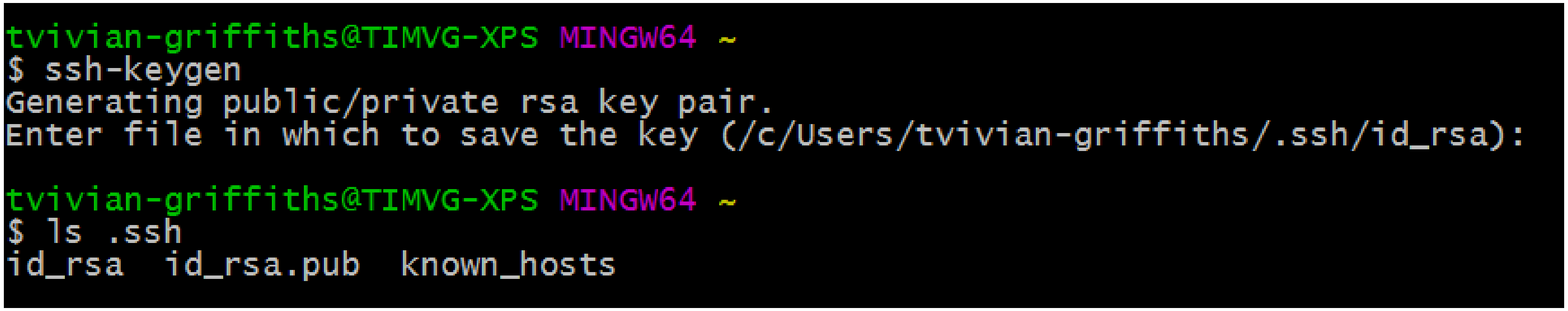
Adnan Fiaz

afiaz@mango-solutions.com

Acknowledgement/Inspiration

- https://lennerd.github.io/git-for-beginners
- http://happygitwithr.com/
- https://speakerdeck.com/alicebartlett/git-for-humans
- https://suzan.rbind.io/2018/03/reflections-4-months-of-github/
- https://www.atlassian.com/git/tutorials/git-stash
- http://r-bio.github.io/intro-git-rstudio/

Create account and add ssh key



```
tvivian-griffiths@TIMVG-XPS MINGW64 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/tvivian-griffiths/.ssh/id_rsa):

tvivian-griffiths@TIMVG-XPS MINGW64 ~
$ ls .ssh
id_rsa  id_rsa.pub  known_hosts
```

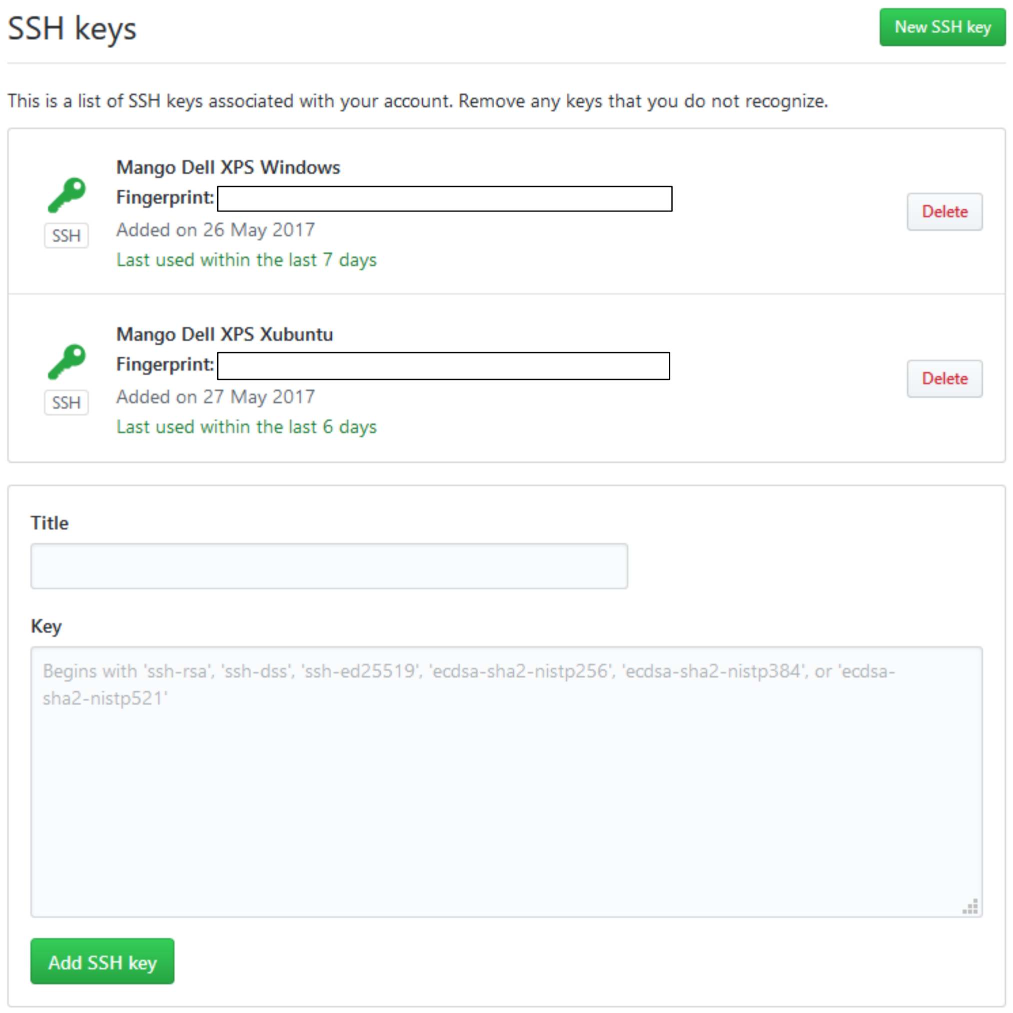
- The password is optional
- If set, you need to enter it for all connections with GitHub
- I’m not using it for this presentation
- You can always set another one later

Add the key to GitHub

- Click profile drop down on top right of the GitHub nav-bar
  - Select **Settings**
  - **SSH and GPG keys**
  - **New SSH key** Green button
  - Provide a Title
  - In Terminal, from home directory **`(cd)`** type:

    ```
    $ cat .ssh/id_rsa.pub
    ```
  - Copy and paste into **Key** field
  - **CTRL-Insert** to copy in Git Bash

Add the key to GitHub



SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Mango Dell XPS Windows
Fingerprint: [redacted]
Added on 26 May 2017
Last used within the last 7 days

Mango Dell XPS Xubuntu
Fingerprint: [redacted]
Added on 27 May 2017
Last used within the last 6 days

Title

Key

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'