Git: Collaborative coding

How do we work with others?

Jo Hallett

October 20, 2025

Last time

- We introduced Git
- We showed you how to make a commit
- We talked about remotes
- We mentioned branches

This time

- How do we collaborate with others?
- How do we fetch other people's changes?
- How do we send them our own changes?

I've cloned a repo!

```
$ git status
```

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

```
$ git remote -v
```

```
origin /home/jo/Repos/Talks/COMS10012-Software-Tools/2024/05-git/Origin-Counting/ (fetch)
origin /home/jo/Repos/Talks/COMS10012-Software-Tools/2024/05-git/Origin-Counting/ (push)
```

```
$ git log --oneline
```

```
4e8e34a Adds 3
46c4069 Adds 2
cc36517 Adds 1
```

Lets fetch

Last time when we fetched there was nothing new…

```
$ git fetch
```

```
From /home/jo/Repos/Talks/COMS10012-Software-Tools/2024/05-git/Origin-Counting
   4e8e34a..582d983 main   -> origin/main
```

```
$ git status
```

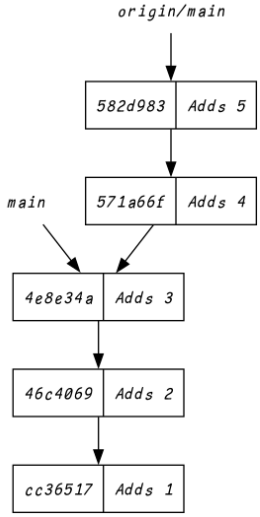
```
On branch main
Your branch is behind 'origin/main' by 2 commits, and can be fast-forwarded.
  (use "git␣pull" to update your local branch)

nothing to commit, working tree clean
```

```
$ git log --remotes --oneline
```

```
582d983 Adds 5
571a66f Adds 4
4e8e34a Adds 3
46c4069 Adds 2
cc36517 Adds 1
```

What's going on?



We want to update `main` to include the work in `origin/main`.

```
$ git merge origin/main
```

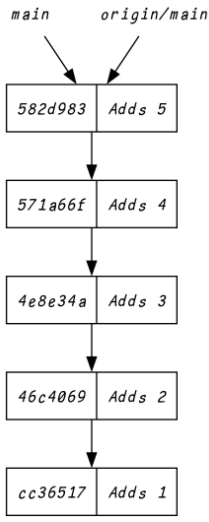
```
Updating 4e8e34a..582d983
Fast-forward
 4 | 0
 5 | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 4
 create mode 100644 5
```

```
$ git status
```

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

And now



And now our graph looks correct!

- …but Git advised us earlier to run `git pull`
- We ran `git fetch` and `git merge`?

Git commands are built on other commands

- The porcelain commands are those that are for people
- The plumbing commands are those for building porcelains with
- Over time the distinction has become blurred!

`git pull` does the same as a `git fetch` and a `git merge`[a]

- There are a lot of commands like this

[a] Usually, it can also do a `rebase` instead of a merge if you prefer that, but we're getting ahead of ourselves.

Collaboration

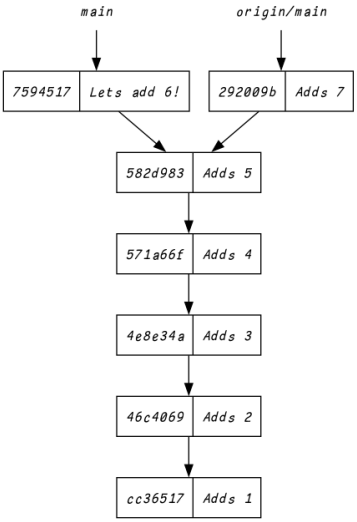
In the last example, the only difference between our two histories was that there was more work on the remote.

- Git could bring `main` up to `origin/main` just by fast-forward-ing it through the history

What happens if we've also done some work?

- Lets pretend we're working with someone else…
- They're gonna work on adding 7… we're gonna work on adding 6…

What's going on?




```
$ git fetch
```

```
582d983..292009b main    -> origin/main
```

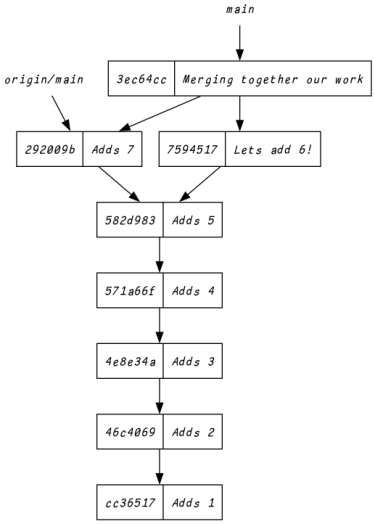
```
$ git status
```

```
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git␣pull" if you want to integrate the remote bra

nothing to commit, working tree clean
```

```
$ git merge -m "Merging␣together␣our␣work"
```

And now?



Now our tree looks like this!

- But there's a problem!

Sending our changes out

The origin doesn't know about our merge!

- We need to send our changes up to it

```
$ git push
```

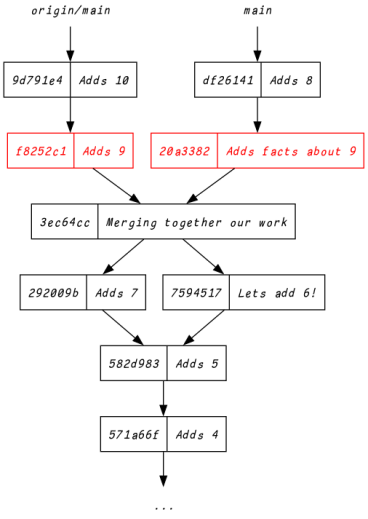
```
To /home/jo/Repos/Talks/COMS10012-Software-Tools/2024/05-git/Origin-Counting/
   292009b..3ec64cc main -> main
```

This does not change our collaborators' code tree!

- No one can do work on a remote repo directly[1]
- They need to run `git pull` to `fetch` and `merge` the changes in their local copy

[1] Technically they're called bare repos and are basically the contents of the invisible `.git/` folder. Create them with `git clone --bare` and read the Git book.

Lets keep going!




```
$ git pull
```

```
Auto-merging 9
CONFLICT (add/add): Merge conflict in 9
Recorded preimage for '9'
Automatic merge failed; fix conflicts and then commit the
```

```
$ git status
```

```
On branch main
Your branch and 'origin/main' have diverged,
and have 4 and 3 different commits each, respectively.
  (use "git␣pull" if you want to integrate the remote bra

You have unmerged paths.
  (fix conflicts and run "git␣commit")
  (use "git␣merge␣--abort" to abort the merge)

Changes to be committed:
        new file: 10

Unmerged paths:
  (use "git␣add␣<file>..." to mark resolution)
        both added:   9
```

A merge conflict!

Inside 9 we'll see:

```
<<<<<<< HEAD
Nine is semi-prime
Nine is the biggest single digit number
=======
This is 9!
It is a square number.
It looks a bit like a 6

>>>>>>> 5422771c6ceee14c9758c3073f97f43f9aa92244
```

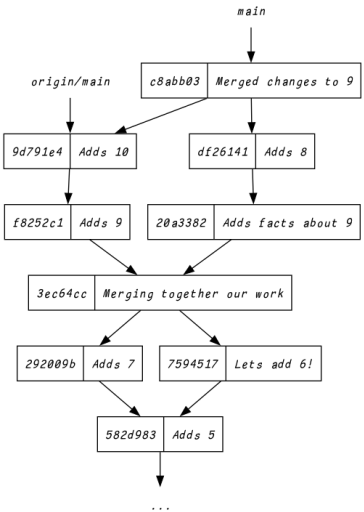
`<<< HEAD` to the equals Stuff on your end

equals to the `>>> id` Stuff on the remote end

Your job is now to edit it back to being correct!

- (A good merge tool like Meld (or Emacs) really helps!)

Fix it



This is 9!
It is a square, semi-prime number.
It looks a bit like a 6

```
$ git add 9
$ git commit -m "Merged changes to 9"
```

```
[main c8abb03] Merged changes to 9
```

Messy

Some people really don't like the merge commits…

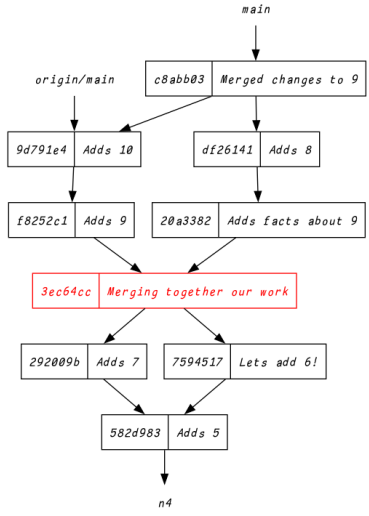
- They think they look messy
- Not the way older version controls did it

Wouldn't it be neater if instead of merging the work we rewrote the history so it was done later?

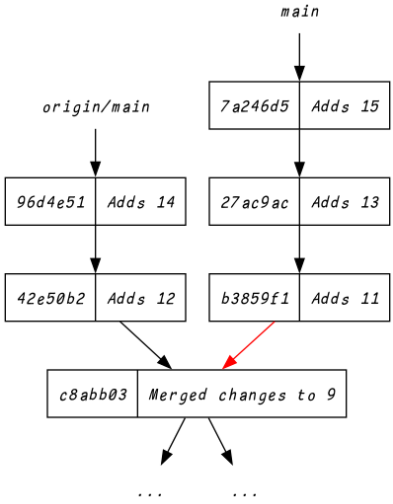
- Then we keep a nice straight line?!

The command you want for this is `git rebase`

- Here be dragons

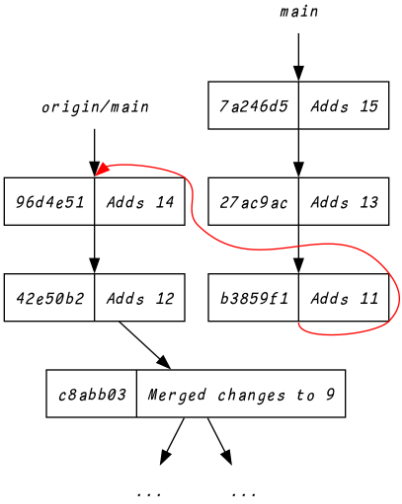


Lets rebase!



We want to cut this edge between `b3859f1` and `c8abb03` and move it…

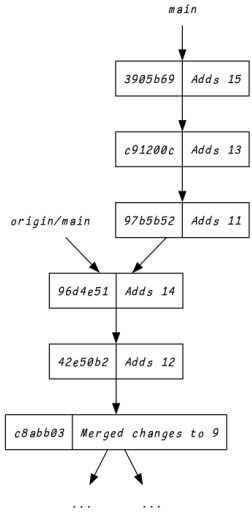
Threading the needle



And reattach it up here

- Then you should just be able to fast-forward `origin/main`
- No need for a merge!

History rewritten!




```
git rebase --onto origin/main
```

```
Successfully rebased and updated refs/heads/main.
```

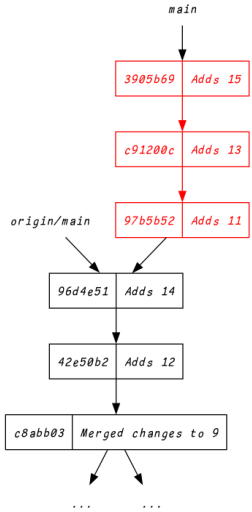
The ids of the rebased commits have changed!

- Git commits id's are based off their own data…
- And the commit before them…

If you prefer this approach to merging

- `git pull --rebase`
- Or set it as the default
- Do whichever your boss tells you

Still messy



Do we really need one commit per file?

- Seems like a lot of noise?

More normally you'd see this when hacking about

- Did some work
- Did some more work
- Argh that last commit had a mistake
- Fxied the mistayk
- …its Friday and I'm tired

Again we can fix this with `git rebase`

Interactive rebasing

```
git rebase -i origin/main
```

And it will kick you into your text editor…

```
pick 97b5b52 Adds 11
pick c91200c Adds 13
pick 3905b69 Adds 15

# Rebase 96d4e51..3905b69 onto 96d4e51 (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                   commit's log message, unless -C is used, in which case
#                   keep only this commit's message; -c is same as -C but
#                   opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
#        create a merge commit using the original merge commit's
#        message (or the oneline, if no original merge commit was
#        specified); use -c <commit> to reword the commit message
# u, update-ref <ref> = track a placeholder for the <ref> to be updated
```

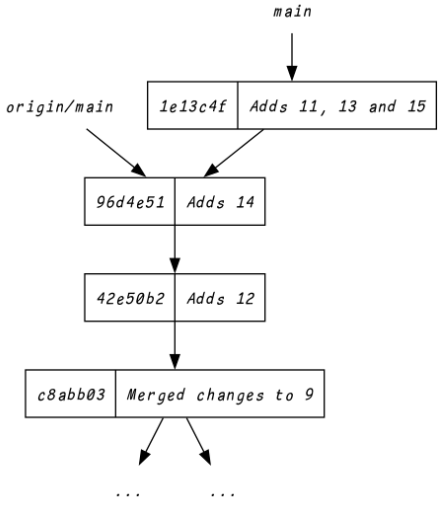
Edit the rebase script

Save and quit when done…

```
r 97b5b52 Adds 11
f c91200c Adds 13
f 3905b69 Adds 15

# Rebase 96d4e51..3905b69 onto 96d4e51 (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                    commit's log message, unless -C is used, in which case
#                    keep only this commit's message; -c is same as -C but
#                    opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
#       create a merge commit using the original merge commit's
#       message (or the oneline, if no original merge commit was
#       specified); use -c <commit> to reword the commit message
# u, update-ref <ref> = track a placeholder for the <ref> to be updated
#                       to this position in the new commits. The <ref> is
#                       updated at the end of the rebase
#
```

Neater



Much neater!

- Rebasing like this to tidy your commits up is a professional curtesy.
- People will think less of you if you don't
- Companies and open source maintainers will probably make you

Why not go further?

- Why not rebase and squash a whole lot more commits?

The one thing I hate is talking to people…

If you go beyond what has already been pushed

- Git won't let you push again, because it looks like work is being lost
- If you run `git push --force` and there isn't any protection it will do it though

At this point all your colleagues need to fix a bunch of stuff when they pull

- Couple of hours `cherry-pick`-ing their work onto your updated tree
- They now hate you
- You owe them beer/blood/money
- You are a **bad person**

Similarly if you push broken code onto the `main` branch

- Any build automation tools will fail
- Your colleagues now hate you
- You will be made to stay late to fix it
- You are a **bad person**
- You might be fired (if you work for IBM/Google… `s/might/will/`)

Collaborating with strangers

So far we've been dealing with repositories where you can `push` to them.

- If your building code with your friends or colleagues that is fine
- If you want to do opensource work that isn't going to be the case[2]

How do you work with other people when you don't know them?

[2] Usually. FreeBSD will give you a commit-bit if you send them high quality work.

Pull requests

This is the way Github wants you to collaborate.

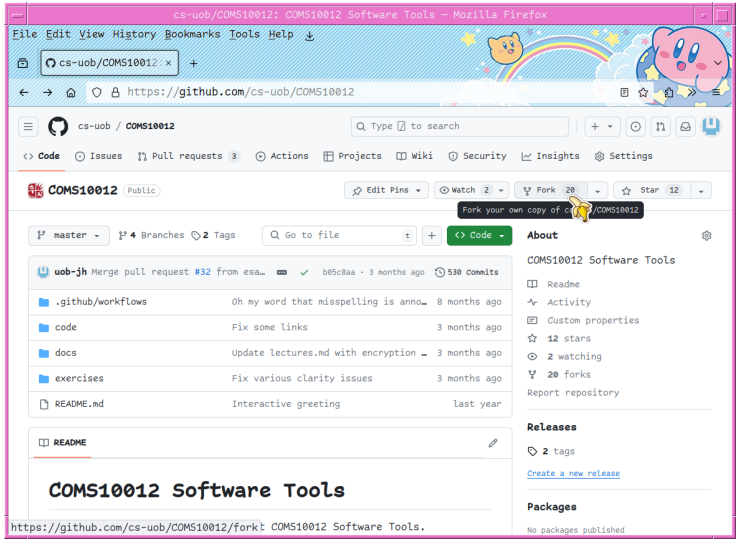
- Very similar process for other forges

The process goes:

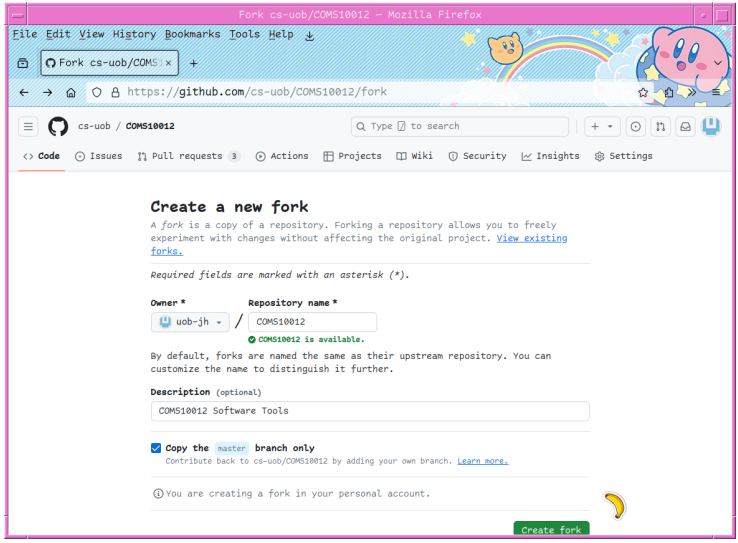
- Clone someone else's repo on the forge
- Do your work
- Send a pull request back to the original repo to merge
- Discuss the changes
- Owner merges maybe?

If you spot a mistake in the slides or labs this is what we'll ask you to do!

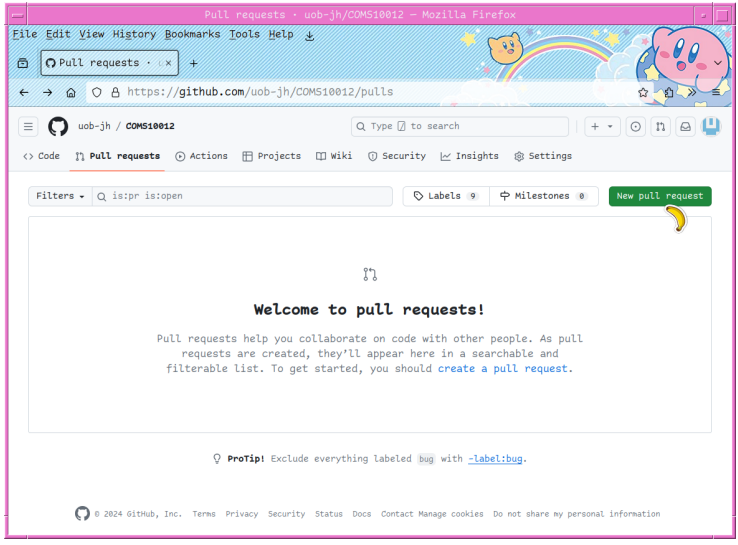
Clone


cs-uob/COMS10012: COMS10012 Software Tools – Mozilla Firefox
File Edit View History Bookmarks Tools Help
cs-uob/COMS10012
https://github.com/cs-uob/COMS10012
cs-uob / COMS10012
Type / to search
Code
Issues
Pull requests 3
Actions
Projects
Wiki
Security
Insights
Settings
COMS10012 Public
Edit Pins
Watch 2
Fork 20
Star 12
Fork your own copy of cs-uob/COMS10012
master
4 Branches
2 Tags
Go to file
Code
uob-jh Merge pull request #32 from esa… b05c8aa · 3 months ago 530 Commits
.github/workflows Oh my word that misspelling is anno… 8 months ago
code Fix some links 3 months ago
docs Update lectures.md with encryption … 3 months ago
exercises Fix various clarity issues 3 months ago
README.md Interactive greeting last year
README
COMS10012 Software Tools
https://github.com/cs-uob/COMS10012/fork COMS10012 Software Tools.
About
COMS10012 Software Tools
Readme
Activity
Custom properties
12 stars
2 watching
20 forks
Report repository
Releases
2 tags
Create a new release
Packages
No packages published

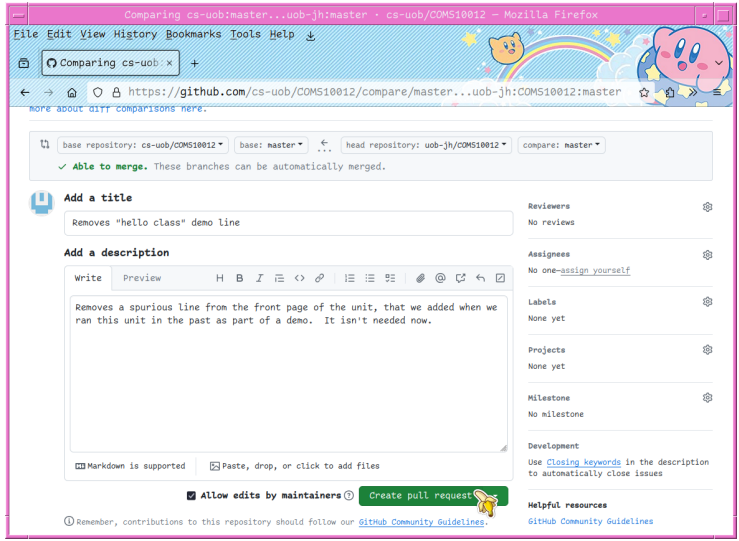
Work


Fork cs-uob/COMS10012 – Mozilla Firefox
File Edit View History Bookmarks Tools Help
Fork cs-uob/COMS
https://github.com/cs-uob/COMS10012/fork
cs-uob / COMS10012
Type / to search
Code
Issues
Pull requests 3
Actions
Projects
Wiki
Security
Insights
Settings
Create a new fork
A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. View existing forks.
Required fields are marked with an asterisk (*).
Owner *
Repository name *
uob-jh
/
COMS10012
COMS10012 is available.
By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.
Description (optional)
COMS10012 Software Tools
Copy the master branch only
Contribute back to cs-uob/COMS10012 by adding your own branch. Learn more.
You are creating a fork in your personal account.
Create fork

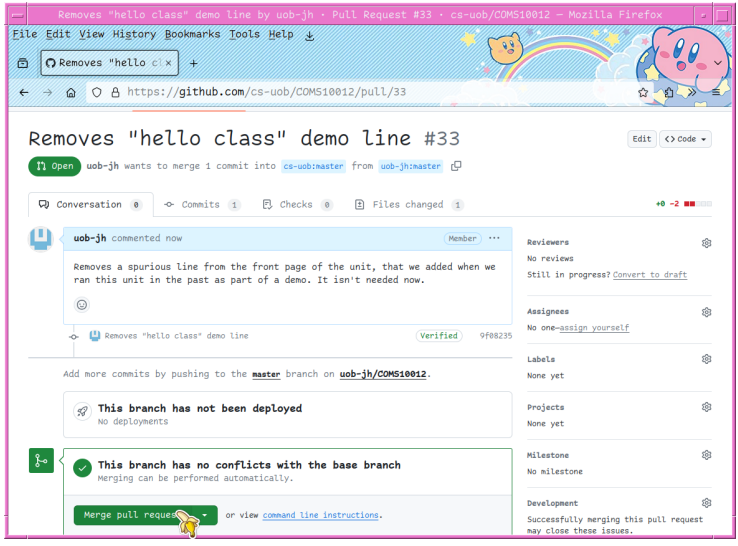
Pull?


Pull requests · uob-jh/COMS10012 — Mozilla Firefox
File Edit View History Bookmarks Tools Help
Pull requests ·
https://github.com/uob-jh/COMS10012/pulls
uob-jh / COMS10012
Type / to search
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Filters
is:pr is:open
Labels 9
Milestones 0
New pull request
Welcome to pull requests!
Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should create a pull request.
ProTip! Exclude everything labeled bug with -label:bug.
© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Discuss


Comparing cs-uob:master...uob-jh:master · cs-uob/COMS10012 — Mozilla Firefox
File Edit View History Bookmarks Tools Help
Comparing cs-uob
https://github.com/cs-uob/COMS10012/compare/master...uob-jh:COMS10012:master
more about diff comparisons here.
base repository: cs-uob/COMS10012
base: master
head repository: uob-jh/COMS10012
compare: master
Able to merge. These branches can be automatically merged.
Add a title
Removes "hello class" demo line
Add a description
Write
Preview
Removes a spurious line from the front page of the unit, that we added when we ran this unit in the past as part of a demo. It isn't needed now.
Markdown is supported
Paste, drop, or click to add files
Allow edits by maintainers
Create pull request
Remember, contributions to this repository should follow our GitHub Community Guidelines.
Reviewers
No reviews
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Development
Use Closing keywords in the description to automatically close issues
Helpful resources
GitHub Community Guidelines

Merge


Removes "hello class" demo line by uob-jh · Pull Request #33 · cs-uob/COMS10012 — Mozilla Firefox
File Edit View History Bookmarks Tools Help
Removes "hello c
https://github.com/cs-uob/COMS10012/pull/33
Removes "hello class" demo line #33
Edit
Code
Open
uob-jh wants to merge 1 commit into cs-uob:master from uob-jh:master
Conversation 0
Commits 1
Checks 0
Files changed 1
+0 −2
uob-jh commented now
Member
Removes a spurious line from the front page of the unit, that we added when we ran this unit in the past as part of a demo. It isn't needed now.
Removes "hello class" demo line
Verified
9f08235
Add more commits by pushing to the master branch on uob-jh/COMS10012.
This branch has not been deployed
No deployments
This branch has no conflicts with the base branch
Merging can be performed automatically.
Merge pull request
or view command line instructions.
Reviewers
No reviews
Still in progress? Convert to draft
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Development
Successfully merging this pull request may close these issues.

Not everyone uses Github

Git $\neq$ Github

Not everyone uses forges

- Especially since Github is owned my Microsoft
- Using GUIs is clunky (if you're quick with a commandline)

Git's default way of sharing changes is by emailing patches

- Kinda old skool now, but can be really powerful

Sending patches

```
$ git status
```

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

nothing to commit, working tree clean
```

```
$ git format-patch origin/main
```

```
0001-Adds-note-about-sending-pull-requests.patch
```

Patch files

```
From 8a955e579d64b82dd7c5ae832e3ca88f36d24a83 Mon Sep 17 00:00:00 2001
From: Jo Hallett <bogwonch@bogwonch.net>
Date: Mon, 15 Jul 2024 13:56:52 +0100
Subject: [PATCH] Adds note about sending pull requests

---
 README.md | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

diff --git a/README.md b/README.md
index 4e549e0..af9bb04 100644
--- a/README.md
+++ b/README.md
@@ -4,4 +4,6 @@ This is the repository for the unit COMS10012 Software Tools.
 If you are looking for the unit website, it is at https://cs-uob.github.io/COMS10012.
-To clone this repository to your computer, type `git clone https://github.com/cs-uob/COMS10012` in a terminal. This repository is public, so you do not need an account
+To clone this repository to your computer, type `git clone https://github.com/cs-uob/COMS10012` in a terminal. This repository is public, so you do not need an account
+
+If you spot a mistake send us a pull request!
```

Applying patch files

```
$ git apply 0001-Adds-note-about-sending-pull-requests.patch
```

```
$ 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: README.md

Untracked files:
  (use "git␣add␣<file>..." to include in what will be committed)
        0001-Adds-note-about-sending-pull-requests.patch

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

Or if you wanna go fast…

```
$ git am 0001-Adds-note-about-sending-pull-requests.patch
```

```
Applying: Adds note about sending pull requests
```

```
$ git log --oneline
```

```
c8cd974 Adds note about sending pull requests
db70366 Build and deploy mdbook
6abb5d0 Merge pull request #33 from uob-jh/main
9f08235 Removes "hello␣class" demo line
```

(Checkout `git send-email` to automate the patch sending process ;-))

That's the basics…

I know this is a lot to take in but that's the basics

- The **only** way to get comfortable with this is to actually do it
- …see you in the lab ;-)

Bonus

As you use Git more and more, little things are going to start to annoy you.
If you compile code you'll end up with a load of object files (e.g. `.o` or `.class`) around

- You don't want to add these to Git.
- Every time you recompile they'll change.
- If someone needs them they can recompile but they won't usually work on their system unmodified

If you work with Mac user's they will eventually commit a `.DS_Store` file

- What even are they?

We would like Git to ignore all of these files…

.gitignore

At the root of your repo, you can create a file called `.gitignore`

- If you add this file (or commit it) then everyting it mentions will be ignored

```
*.class
*.o
.DS_Store
build/
!build/README.txt
```

Git repos for ignoring git files

If you go to https://github.com/github/gitignore

- You can find a huge list for every programming language under the sun

```
# Compiled class file
*.class

# Log file
*.log

# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar
```

What if I want to apply these everywhere?

Setting a `.gitignore` per repo is pretty useful

- But what if you want to always ignore certain files?

You can set:

```
$ git config --global core.excludesFile
```

But this will just apply to your machine

- Per repo `.gitignore` will get sent to contributors too
- So good for editor/OS specific ignores, less good for repo specific ones

That's all, folks!

We talked about:

- Merging branches
- Dealing with conflicts
- Rebasing
- Github pull requests
- Sending patches
- Git ignores