Wallace Lab Git Workshop

Flic Anderson & Sam Haynes
26/11/2020

HOW TO (RE-)USE THIS MATERIAL

This is a `.html` presentation created in `R Markdown` with `ioslides`.

(It's been written in a .Rmd file, and we generated .html slides by 'knitting' it in Rstudio.)

Check out the code used to make these slides at the Workshop repo on Github.

To view the slides from that repo: view the PDF file, or download the raw html file (or clone the whole repo!) locally & open in a browser.

You can also adapt these slides for your own presentations if you like - we've got a MIT Licence on the repo, which means:

"Basically, you can do whatever you want as long as you include the original copyright and license notice in any copy of the software/source." Source: tl;drLegal

Outline:

Outline:

- Intro to Git, GitHub
- The Git Workflow
- Favourite Papers Lab Website Task
- Issue Tickets, Bells & Whistles
- Branching
- Writing Papers
- Troubleshooting
- Questions

Intro:

Git

Git is a *version control system*.

- It doesn't copy your files, as much as keep track of all `changes` ever since you told git to `add` the file to its logs
- Best for `text` files; can't see 'into' binaries (e.g. pdfs, pics)
- Saves the changes you tell it to `commit` to its log in project-specific folders referred to as a repository, or `repo`
- Very useful: undo/redo, backups, switching computers, safely 'experimenting'

It can be used via *Command Line* (the terminal!) or via a *Graphical User Interface* (GUI), on pretty much all systems.

GitHub

GitHub is an *online platform* for storing your git repository. There are others (e.g. GitLab, Bitbucket).

- Keep track of all your repos online - `public` or `private`
- You can tell git to `push` its list of your file changes up to GitHub.
- Or `pull` changes down (yours or someone else's!) and `merge` them with your local copies.
- Contribute to others' work via a `pull request`

Git Workflow

Fork

Clone

Branch

Edit

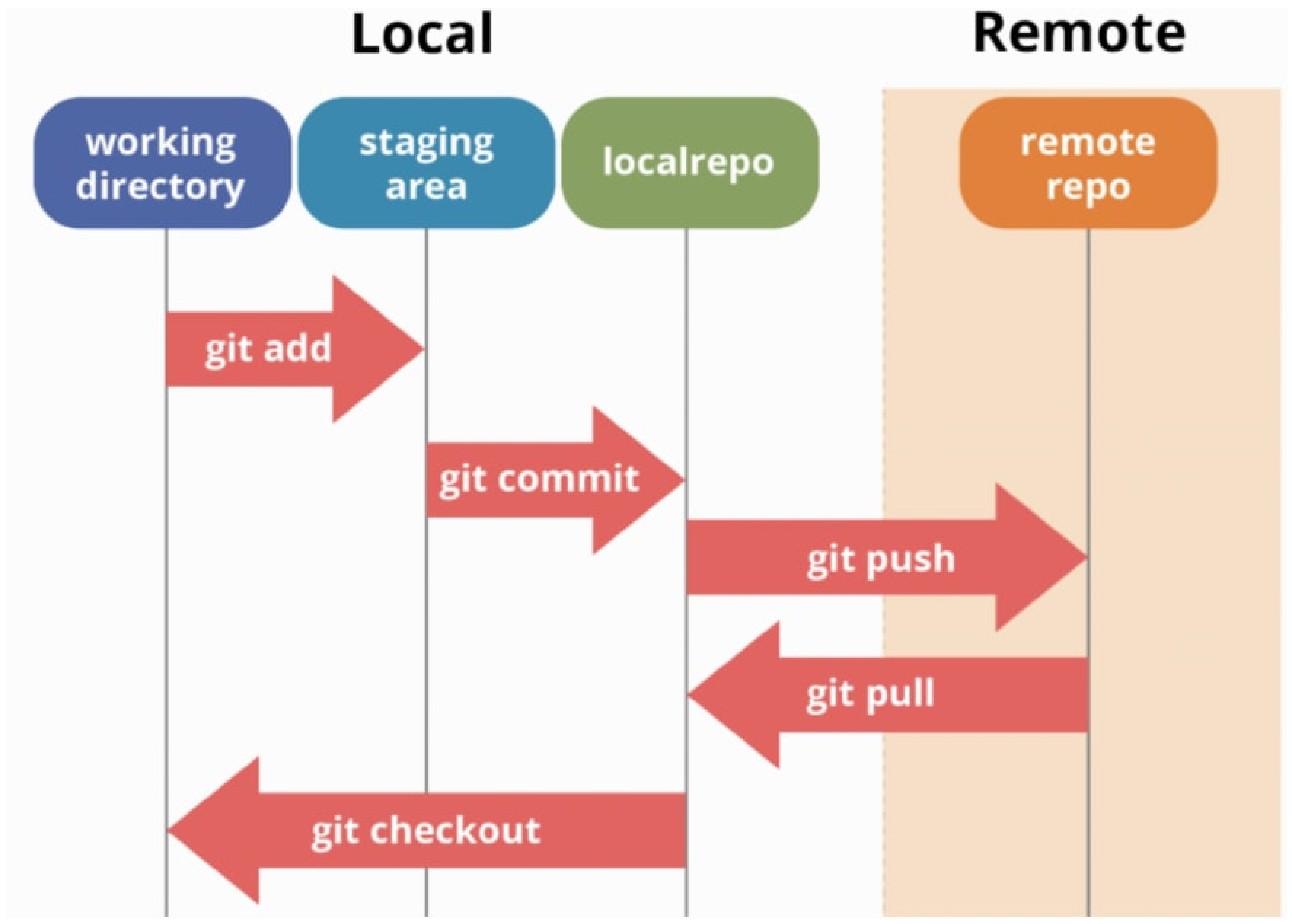
Add

Commit

Push

Pull Request

Git Workflow



How To Write A Good Commit Message

```
git commit -m "..."
```

Be consise & consistent!

- WHAT did you do (e.g. fix, rework, add, update)
- WHERE (e.g. filename, function name, section of paper, variable)

```
git commit -m "add Flic's favourite paper to '2020-11-26-
favourite-papers.md'"
```

Favourite Papers Task

Task: “Add your favourite paper & 1-line summary to a blog post on the lab website”

Aim: practise the basics of fork/branch/change/PR et & make an interesting blog post!

The Wallace lab website is built from a github repository, and uses GitHub Pages to build the website HTML/CSS from the files in a GitHub repository & hosts them online.

Our task is to add favourite-paper info to the blog post file using the git workflow so we can publish it to the website.

Flic’s Task Setup

Fork, Clone, Branch, Edit, Add, Commit, Push, Pull Request

- `forked` (made a copy of) Edward’s website repo on GitHub
- `cloned` the forked repository (downloaded the github copy to my local computer)
- made a `branch` for my changes called “workshop-blogpost”
- created and `edited` a blog post file: “blog/_posts/2020-11-26-favourite-papers.md”
- `added` this file to my ‘staging area’ locally
- written a `commit` message to explain my changes
- `pushed` my changes up to my copy of the repo on GitHub
- (no `pull request` yet, we’re ready to collaborate!)

Hang On A Minute!

Are You Already Forked?

If you already have a fork of the lab website, there's an extra step to make sure you've got the latest changes in Edward's original repo!

I made a helpsheet!

Bitly Link

New To Forking?

Fear not! There’s a first time for all of us.

Here’s how to get started!

[Bitly Link bit.ly/fave-papers-forking](bit.ly/fave-papers-forking)

Favourite Papers Task

Favourite Papers Steps

Fork, Clone, Branch, Edit, Add, Commit, Push, Pull Request

- fork the website in github (it copies all branches)
- clone (download) the fork to your local computer
- make branch (e.g.”sams-favourite-paper”)
- make edits to “blog/_posts/2020-11-26-favourite-papers.md”
- git add blog file
- git commit blog file with helpful message
- git push to your github (to remote)
- check on github the changes are there
- make pull request to Edward’s repo (check it’s “staging” branch)

Favourite Papers Steps

Fork, Clone, Branch, Edit, Add, Commit, Push, Pull Request

Bitly Link: bit.ly/favourite-papers-task

Favourite Papers Next Steps

Check Pull Request & Merge

Edward checks the blog post & merges the changes to `master` branch … publishing the blogpost!

PARTY!

- http://nedroid.com/2009/05/party-cat-full-series/

Issue Tickets

Issue Tickets

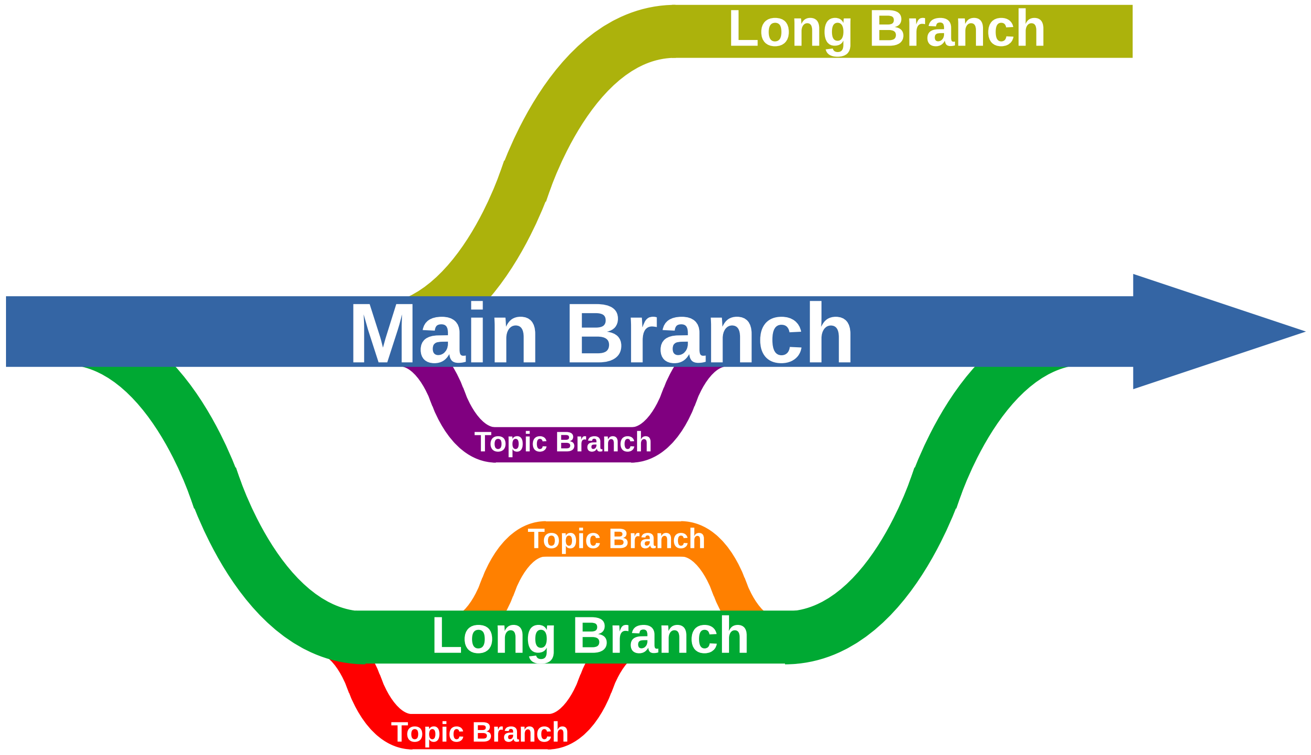
- GitHub tool: create ‘issues’ within a repo, where you can add notes.
- markdown format - can include code snippets, formatting, links, emojis!
- @-tag in other GitHub users (“@FlicAnderson, do you get this 404 error before?”)
- assign labels to issues like “help wanted”, “bug”, “question”.
- link issues with pull requests, or keep them separate.
- once the issue is solved, you can mark it as closed!

How To Write A Good Issue Ticket

Give a clear and concise description of the problem or new feature to help others get up to speed quickly

- how you found the issue & what you’ve done already to try fix/investigate it/ideas
- tag in colleagues who might offer insights, or answer any questions
- remember to update the issue regularly as progress happens!
- link to other issues in that repo using the hashtag & issue number e.g. #42
- add Add the most useful labels / create your own
- check with people before randomly assigning them issues! :)

Branching


Long Branch
Main Branch
Topic Branch
Topic Branch
Long Branch
Topic Branch

Branching

The concept of branching is a combination of good coding practice and good project management.

It consists of separating projects into larger objectives and smaller, manageable tasks.

It also enables you to prototype changes in isolation, minimising risk, and provides checkpoints to quickly return to when things do go wrong.

Branching

Long vs short

Long Branch

Long branches can exist for months or are even permanent features of a repo.

They can be merged into the main branch multiple times, possibly containing updates on thousands of lines of code.

They are used to isolate large tasks or allow for gate-keeping for projects with continuous updates.

Examples: Staging branch, prototyping branch, software release branch, manuscript chapter branch

Branching

Long vs short

Short (or Topic) Branch

Short branches exist for hours or weeks, they may consist of only one commit.

Once merged with the main branch or a long branch they are deleted.

They enclose small changes in code and separate long term goals into short term tasks.

Examples: issue branch, function branch, figure branch, documentation branch

Branching

Commits vs branches

Commits are checkpoints for sub-tasks, often no more than 10 lines of code.

Topic branches are checkpoints for tasks, with one or more commit.

```
git checkout -b quality_control

git add check_microarray_data.R

git commit -m "function to create
tibble from microarray data"

git commit -am "ggplot functions to display raw data"
```

Branching

Where am I?

```
git branch
```

```
  master
```

```
git branch -a
```

```
  master
  remotes/upstream/master
  remotes/upstream/staging
```

Branching

Local vs remote

GitHub is an amazing collaborative tool, but significantly complicates code management.

Branches and commits will appear on the remote repo that are not on your local repo (and vice versa)

This multiplies again if you originally forked from someone else!

Branching

Tips for code hygiene

- Don’t commit unfinished code and merge incomplete branches
- Update your main branch everyday (typically using git rebase)
- Keep topic branches short and merge-delete often!
- Ensure commits are made in relevant branches

Troubleshooting

Troubleshooting

Stashing

```
git stash [push]
```

Reverts the code back to the last commit and saves changes in temp file.

Perfect for when you realise you’re working in the wrong branch.

Swap to right branch and pop the changes back

```
git checkout other_branch
git stash pop
```

Troubleshooting

Reverting and cherrypicking

```
git revert e8cc44d3ce4b01fb2d211bffec0c69cbaa0d80f4
```

Eventually you will accidentally commit a bug into your branch and need to remove it.

```
git cherry-pick e8cc44d3ce4b01fb2d211bffec0c69cbaa0d80f4
```

Some cases my require you to select exactly which commits to merge (rather than a whole branch)

Troubleshooting

Helpful guides

Git cheatsheet

https://education.github.com/git-cheat-sheet-education.pdf

Git documentation

https://git-scm.com/doc

GitHub Guides

https://guides.github.com/

Questions