Introduction to git and GitHub

Evan Thomas

Research Computing

Goals for today

- Part 1: What is git and what is it for?
- Part 2: Basic git workflows
- Part 3: GUI Clients
- Part 4: Overview of advanced workflows

Create a GitHub account

Please create a GitHub account:

https://github.com/

Requirements

- WiFi access
- Mac Terminal: requires command line tools from managed software centre
- Windows git client
- ssh/putty into unix500: `module load git`

Slide Respository

- A copy of these slides and sample code is available at: `https://github.com/WEHI-ResearchComputing/GitIntro`s
- Send me an email: `thomas.e@wehi.edu.au`
- Visit our web site: `https://rc.wehi.edu.au`

Part 1: What problem(s) does git solve?

- How can you keep track of the changes to your software?
  - Who did what and when?
  - Undo on steroids
- How can several people work on the same software without interfering with other?
- How can those people work at different locations without access to the same filesystems?
- How can you do this with projects consisting of tens of millions of lines of code?
  - fast
  - robust
- Keeps your code safe

Part 1: What is git?

- Written by Linux Torvalds - the creator of linux
- Provides concurrent version control
- Supports a distributed workflow
- Very robust
- Very fast
- Used by the overwhelming vast majority open source projects
- Excellent support for:
    - integration with development tools
    - build and test systems
    - deployment systems
    - hosting and publishing software
- It's free and can be installed pretty much anywhere
- Language, build system, IDE, etc agnostic

Part 2: Gitting started (ha, ha)

You need:

- Some code, or other text based intellectual property
- A local repository, i.e. on your laptop or on a WEHI filesystem
- A remote repository, mostly likely on GitHub

You can create these in any order.

Part 2: Create a remote GitHub repo

1. Login to github
2. Select new repo from the `+` menu
3. fill in details
    1. The name (without spaces)
    2. Description
    3. Optionally select an open source license
    4. Optionally add `.gitignore` for your langauge
    5. add a readme
4. Create the repo

Part 2: Make a local copy

1. Click the green `clone or download` and copy the link
2. Open a terminal, `cd <some convenient directory>`
3. `git clone <paste URL from GitHub>`
4. `cd <repo-name>`
5. `ls -la`

Part 2: The local repo

File	Description
`.git`	A directory containing the entire state and history of the repo
`.gitignore`	descriptions of files that are ignored by git (optional)
`LICENSE`	Text of the license (optional)
`README.md`	Helpful information for people using your source (optional)

Part 2: Basic file workflow: Adding files

- create a file
- Have a look: `git status` - the file is untracked
- `git add <file>` - file is now staged, keep adding files
  - **Note:** If you change a file you need to stage it again!
- Have another look: `git status`
- `git commit –a –m "Create important test"`
- Have another look: `git status`
  - A commit is a point in the history of the repo with a unique identifier
- `git push` (you may need to enter credentials)
- Have another look: `git status`

Part 2: Basic file workflow: Updating your repo

- Edit a file in GitHub
- Have a look: `git status`. Hmm
- `git remote update`
- Have another look: `git status`
- `git pull`, `cat <file>`

Part 2: Resolving conflicts

Scenario: Two people make conflicting changes

- Change the local file and commit
- Change the remote in a different way
- `git pull`
- Merge Conflict!
- Resolve conflict manually
- `git add <file>`
- `git commit -m "..."`
- `git push`

Note: Always pull first!

Part 2: Branching

Create a branch to work on independently

- With a branch you can work independently
  - Not effect other users
  - Maintain the integrity of the main branch
  - Fix defects independently of adding features
- `git branch`
- `git branch exciting-feature`
- `git checkout exciting-feature`
- make some updates
- `git add ...`
- `git commit ...`

PART 2: Pushing a new branch

- Branches initially only exist in the local repo
  - That may be good enough!
- `git push --set-upstream origin exciting-feature`
- This will now be visible:
  - in GitHub
  - in any repos that do a pull

Part 2: Merging branches

- `git branch exciting-feature`
- `git merge master`
  - **Note:** Always merge in the target first!
- Resolve merge conflicts
- `git add ...` ; `git commit ...`
- `git branch master`
- `git merge exciting-feature`
- `git push`

Note: Always merge master into your branch first!

Part 2: Summary of important commands

command	Description
`git clone`	make a local copy of a remote repo
`git status`	current state of the *local* repo
`git add`	add a file to the index, ready to commit
`git commit`	create a new point in the repo's history
`git pull`	update the local repo from the remote
`git push`	update the remote repo from the local
`git branch`	create and view branches
`git checkout`	change the current working branche
`git merge`	merge two branches branches

Part 2: Summary of important commands

command	Description
`git log`	history of commits
`git blame <file>`	who last altered a line in a file (can be misleading)
`git diff`	difference in a file between 2 commits
`git <command> --help`	Get help

Google probably has the answer to your git questions.

Part 2: Be careful

> With great power comes great responsibility - Uncle Ben or maybe Voltaire

Things that can go wrong

- Horrible merge conflicts
  - merge/pull from master frequently
- Loosing modifications by not using `git add`
  - Some IDEs will do this automatically
  - `git status –uno`
- Trying to undo commits
  - `git reset` will discard data and state
- Top tip: `cp –a <repo> <repo>.bup` before trying things.

Part 3: Graphical clients

- Source Tree - available in the Managed Software Centre
- JetBrains IDEs for Python, Javascript, Java, Scala, etc
- Eclipse and derivatives for Java, C++
- Microsoft IDEs
- Xcode

Part 4: Pull requests

- Where collaborators have different levels of ability code may need to be reviewed before incorporated into important branches.
- Pull requests from source branch to a target branch:
    - notify other members of the team that there is branch ready to merge
    - allow team members to review the changes, alter the changes and comment, etc
    - are handled by the by hosting websites like GitHub

Part 4: Contributing to other projects

For example, adding features or fixing defects in popular software

1. Discuss first the owners
2. Fork the respository in the GitHib
3. Bang away
4. Regularly update your fork with the upstream
5. Send a pull request to the owning team
6. Convince them to merge your updates.

That's it!