A short introduction to version control with Git

Git - You'll never learn it all!

Benjamin Blundell[1]

[1]ITS Research
Queen Mary University of London

CIS Software Engineering Day, 2016

Introduction

Me

- b.blundell@qmul.ac.uk
- ITS Research - writing programs for scientists and artists.
- Helping people with their HPC problems.
- Previously ran small company building graphics-type things
- I write stuff at www.section9.co.uk
- I use Git everyday

What is it?

What is git?

- Version control system developed for the Linux Kernel.
- Distributed as opposed to centralised.
- A suite of many, many small tools.

What is github?

- Git, but on the internets.
- User-friendly interface including visualisations.
- A social network of sorts.

How will Git help me right now?

Advantages to version control

- Backing up your work.
- Undo-ing changes you've made.
- Sharing work with others.

Advantages to github

- Collaboration (showing off?).
- Off-site backup of code.
- Public accountability.
- Open-source contribution.
- A place to put all my slides so you can get them easily.

Resources

Reference material

- https://git-scm.com/doc
- https://git-scm.com/book/en/v2
- man pages for git

Tutorials

- https://try.github.io/
- http://gitreal.codeschool.com/

Software

- Git for Windows.
- TortoiseGit.
- Git is built into Visual Studio and others.

Lets begin (0)

Listing 1: setup git for windows

```
L:\Git-2.6.4-32-bit\git-bash.exe
git config --global http.sslverify "false"
```

Git for Windows

- Git for Windows is but one version of git
- Same commands on Linux and MacOS

Lets begin (1)

Listing 2: clone a repository

```
git clone https://github.com/QMUL/gitclass.git
git status
```

Listing 3: create a new repository

```
git init
```

Making changes

Listing 4: Making changes

```
git status
git add <your filename>
git status
```

Differences and Reset

Listing 5: Making changes

```
git diff --staged
```

Listing 6: Reset changes

```
git reset <myfilename>
```

Forking on Github

Forking

- Not strictly a git command per-se. A github.com feature
- Creating our own version and copy online on github
- Related to the original

Remote Copies / Repositories

Remote

- A copy of the repository, complete and somewhere else.
- Could be github, or another directory on the same disk.

Listing 7: Reset changes

```
git remote add cis <address>
git@github.com:MYUSERNAME/gitclass.git
git remote --help
```

Committing changes

commit

- Possibly the most used command
- Staged changes are 'committed'.

Listing 8: git commit

```
git commit --help
git commit -a -m "<my message>"
```

Pushing Commits

push

- Pushing your commits to a remote repository.

Listing 9: git push

```
git push cis master
git diff
```

Checking History

checking

- Looking at the history of commits
- List of git commit messages and unique IDs

Listing 10: git log

```
git log
```

Removing files

Listing 11: Removing Files

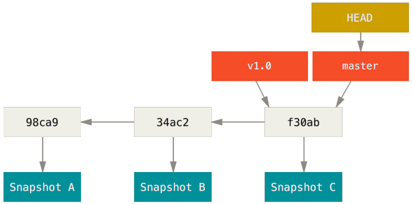
```
git rm <your file name>
git commit -a -m "removed our new file"
```

Its all gone wrong(0)

wrong

- What do we do if we delete or change things and we want to go back
- We need to think about pointers and commit IDs

Pointers



Its all gone wrong(1)

Listing 12: restoring things

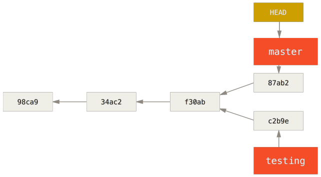
```
git reset --hard HEAD~1
git reset --hard HEAD
```

Branching (0)

branching

- Probably the heart of Git - different code with a common history.
- Can 'branch off' from the main codebase to test things.
- Can merge back at a later date. Everything recorded.

master and testing branches



Branching (1)

Listing 13: branching

```
git branch morespeare
git branch
```

Listing 14: make some changes

```
https://archive.org/details/
  gutenberg?and[]=shakespeare

git commit -a -m "Added more shakespeare to use"
git push origin morespeare
```

Merging

Merging

- The counterpart to branching.
- The trick is to recognise and resolve conflicts

Listing 15: make some changes

```
git checkout master
git merge morespeare
```

Conflicts

Listing 16: possible result of a merge

```
Auto-merging shakespeare_corpus.txt
CONFLICT (content): Merge conflict in shakespeare
Automatic merge failed; fix conflicts and then co
```

Listing 17: a conflict

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.c
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

Collaboration

Using remote repositories with correct access controls we can work collaboratively.

collaboration

- github.com is perhaps the most widely known.
- git-lab.
- gitolite with keys and a server.
- Any remote repository where you can get access.

Advanced Topics

Practical Exercise before we move onto advanced topics.

Stashing

Sometimes you want to temporarily store changes and come back to them later without commiting.

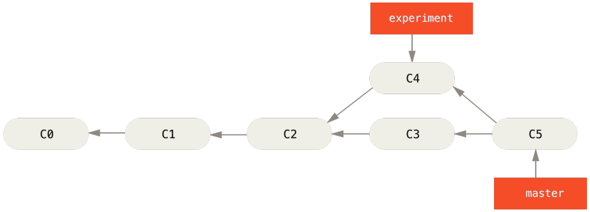
Listing 18: stashing

```
git stash
git status
git stash list
git stash apply
```

rebasing

Rebasing, in some ways, is another way to perform a merge (amongst other things). It *replays* changes on-top of a common ancestor.

example with merging

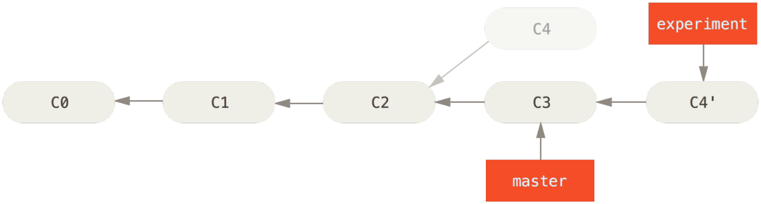


rebasing 2

Listing 19: rebase

```
git checkout experiment
git rebase master
```

example with a rebase

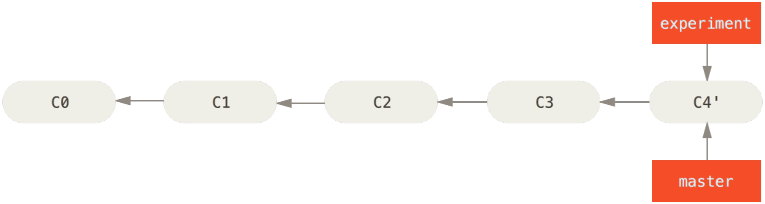


rebasing 3

Listing 20: rebase

```
git checkout master
git merge experiment
```

example with a rebase



tagging

Tags are good ways to mark commits. Also useful for people when they checkout your code.

Listing 21: tagging

```
git tag
git tag –a v1.4 –m "my version 1.4"
git show v1.4
```

Can create lightweight tags (just the checksum) by not adding -a.

diff

Tools to see the differences and create patches from these differences.

Listing 22: diff examples

```
git diff
git diff HEAD
git diff HEAD^ HEAD
```

Various other tools like vimdiff, and more advanced diffs across branches

Listing 23: diff examples 2

```
git difftool --tool=vimdiff --no-prompt \
  origin/togusa:.vimrc .vimrc
```

Flows

A way to organise your work. Use branches and tags to keep work organised.

- development/trunk
- stage/pre-production
- production/live

Integrating Git with workflow

Git and github work well with other tools. Automatic build-tools

- https://travis-ci.org/
- https://jenkins.io/index.html

Integrating Git with workflow 2

Can also automatically test code upon commit ...