Git’er Done

Octocat is dumb

I hate computers

But git is good so use it

My life in the Capra lab

```
 6449556 Sep 30  2017 ancestry_models_old.tar.gz*
 3457642 Nov 30 15:31 final_set_badengfix2.tab.modifications*
 3591576 Nov  9 12:21 final_set_badengfix.tab.modifications*
   24172 Nov 27 13:15 icode_redos.tab*
171462018 Sep 16  2017 old_pdbmap12_nohaplotypes_run.tar.gz*
   55420 Dec 10 12:03 pdb_ancestry_analysis_pdb_engineered_fixedicode.py*
   54242 Nov  1  2017 pdb_ancestry_analysis_pdb_engineered.py*
   54712 Sep 11  2017 pdb_ancestry_analysis.py*
   17790 Dec 17 16:39 pdb_ancestry_bias_v2.py*
    3454 Nov  8 18:36 pdb_mutations.redos.multtrans.tab*
   84579 Nov  1  2017 pdb_mutations.redos.singletrans.tab*
 3415493 Sep 17  2017 pdbs13.tab.modifications*
   98216 Sep 30  2017 single_chain_seqs_old.tar.gz*
27283625 Sep 30  2017 variant_locations_old.tar.gz*
sers/sliwosgr/pdb_ancestry_bias/tmp% oh my god
```

Multiple input datasets

Tons of script versions and bug-fixes

Old output files

Long README about how different output versions were merged

```
Affected files = *fixed* and .modifications

pdb_mutations.redos.singletrans.tab = list of PDB single transcript chains that had to be rerun

pdb_mutations.redos.singletrans.tab = list of PDB multiple transcript chains that had to be rerun

Single transcript reruns were split over 10 sets

1. For each set, remove the lines with 'Column' and 'Index' that were created in bug in code that dumps empty frames when no vars exist.
2. For each set except 1, skip the header line
3. Concatenate all sets 5751 unique chains
    This aligns with the counts in singletrans.tab (5638) + multtrans.tab (114)
4. Create filterall.ls that is the entries in list format that matches the data files.
5. filter the redos from the final datasets with grep -vf filterall.ls

Note: these 12 are in the redo set and not in the previous final dataset:
These are all titin. I added them to a filterttn.ls to filter from the redo set so that the chain counts could be the same between chain and fix
dont show up in the initial Final set.

1G1C A
1G1C B
1WAA A
1WAA B
1WAA C
1WAA D
1WAA E
1WAA F
1YA5 T
2F8V T
2F8V Y
2RQ8 A

The filtered redo sets have suffix _nottn

*I'm not sure why I left the header from set 1 as I need to strip that anyway before appending to the old final dataset...

Ok, now the suffix is _nohead_nottn_nohead...

6. Append the redos (after concatenating sets, removing titin, and removing headers) to the end of the filtered final sets.
(Kept the original datasets with the redos filtered out in tmp/ for now)

7. Do the same for the .modifications file as all the *fixed*

This is a mess, you are a mess
```

My life in the Meiler lab…

Hey Greg model these loops now! No more excuses!

Sure Jens no problem I love modeling protein loops!

My life in the Meiler lab chapter 2

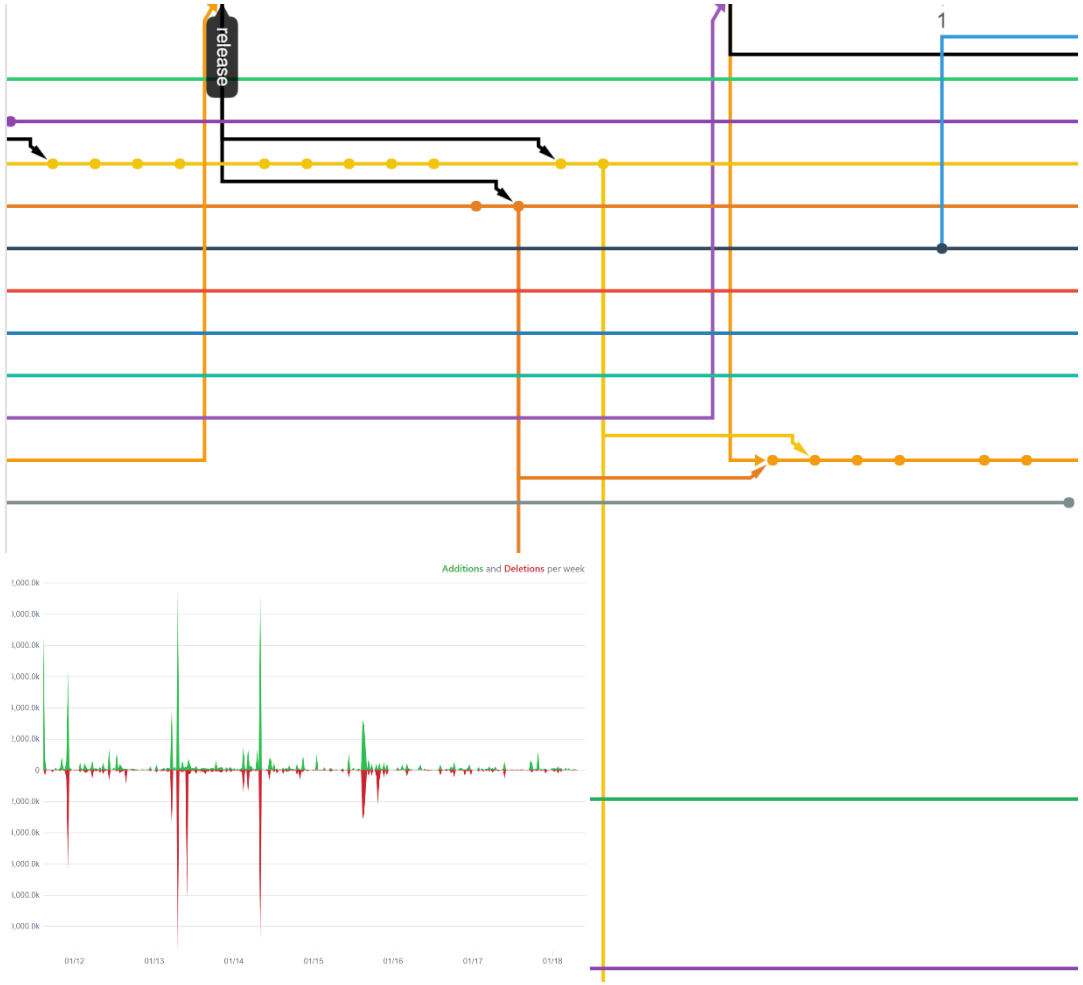
This loop has a disulfide bond!

Hahahaha someone refactored loop building and it doesn't care about your stupid disulfide bond hahahahahhaah EAT IT

Rosetta Commons

I like fixing code it's… OH MY GOD

1 week



April 26, 2018 – May 3, 2018

Period: 1 week

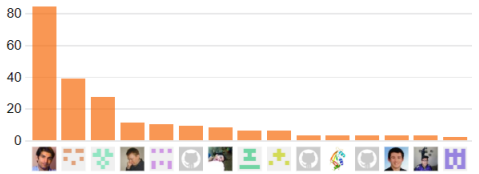
Overview

26 Active Pull Requests

4 Active Issues

18	8	0	4
Merged Pull Requests	Proposed Pull Requests	Closed Issues	New Issues

Excluding merges, **26 authors** have pushed **86 commits** to master and **234 commits** to all branches. On master, **257 files** have changed and there have been **20,352 additions** and **12,555 deletions**.



18 Pull requests merged by **8** people

Merged #3171 Fixing PyRosetta GCC build 5 hours ago

Merged #3170 secstruc strucfrags 13 hours ago

Merged #3158 Revert "Revert "Make auto_setup_metals work with centroid mode."" a day ago

Merged #3156 Incremental merge 7: Trying for some additional speed improvements with the buried_unsatisfied_penalty scoreterm 2 days ago

Merged #3155 Incremental merge 6: Fix some spiky memory use in the buried_unsatisfied_penalty scoreterm 2 days ago

Merged #3142 Remove cached data from suite potential 2 days ago

Merged #3164 Very Small Multistage Rosetta Scripts Update 2 days ago

Git Bisect is magic

- Tell Git that version 1.0 works (begin)
- Tell Git your current version is broken (end)
- Magically, you are teleported to the working directory midway between
- Compile and test
- Tell Git it works!
- Magically you are now ¾ between
- Repeat 30000 times
- Find yourself in identical working directory of bad programmer
- Git tells you exactly what changed before and after

How to fake being a programmer

- Use git bisect
- Identify 1 line of code that broke some feature years ago
- Change that line of code
- Compile
- Show people you fixed something in Rosetta
- Accolades

What is git? (and github?)

- The most commonly used version control system (VCS)
- A historian
  - No more _fixed, _fixed2, _fixed2_final, _fixed2_final_realfinal
- A multitasker
  - Work on multiple things at once with ease

Git

- A cloud
  - Share your code, get code from others, backup code

GitHub

- Everything is easier. **Everything you already do is easier**.
  - Git was made by this guy (in 2005)
  - Yeah he made linux too.
  - But he didn't make github because he's lazy
- Something to put on your resume

Linus Torvalds

What is a Git?

- Quoting Linus: "I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'Git'".
  - ('git' is British slang for "pig headed, think they are always correct, argumentative").

- What Git stands for (according to Linus)
  - "Global information tracker": you're in a good mood, and it actually works for you. Angels sing and light suddenly fills the room.
  - "Goddamn idiotic truckload of sh*t": when it breaks

- The original description
  - This is a stupid (but extremely fast) directory content manager.
    It doesn't do a whole lot, but what it *does* do is track directory contents efficiently. It is intended to be the base of an efficient, distributed source code management system. This package includes rudimentary tools that can be used as a SCM, but you should look elsewhere for tools for ordinary humans layered on top of this.

- Today’s description
  - Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

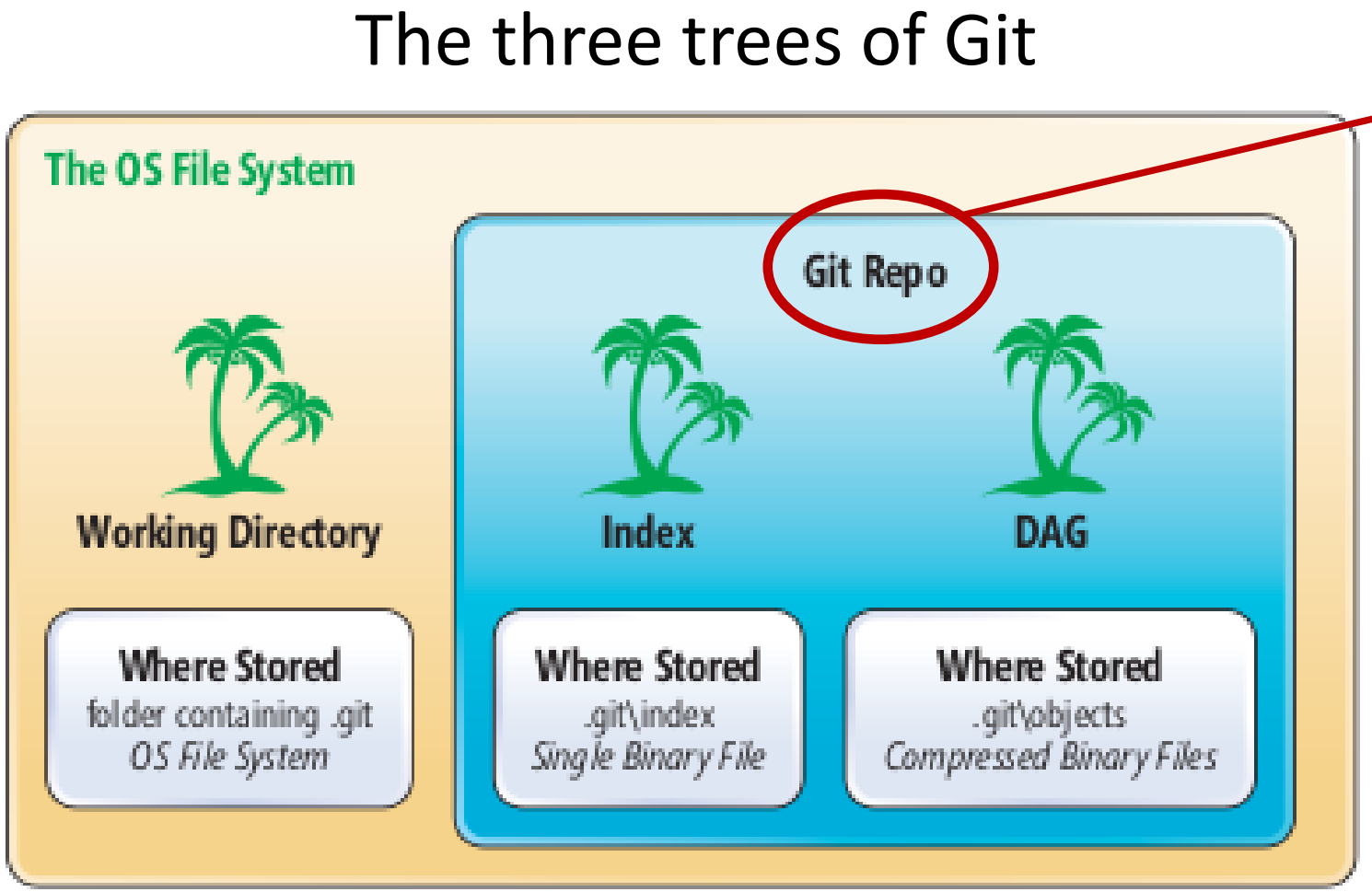
Version Control (VCS)

- Keep track of every modification
  - Go back in time easily
  - Less clutter
  - **Keeping notes is built in** ← Science says this is important
    - Anyone can see what decisions you made and why you made them (my future self is stranger danger)

- Independent work
  - No file locking nonsense
  - Parallel testing and development
  - Experiment without clutter or interruption

- Security
  - Cryptographic keys

Git – a forest with 3 trees



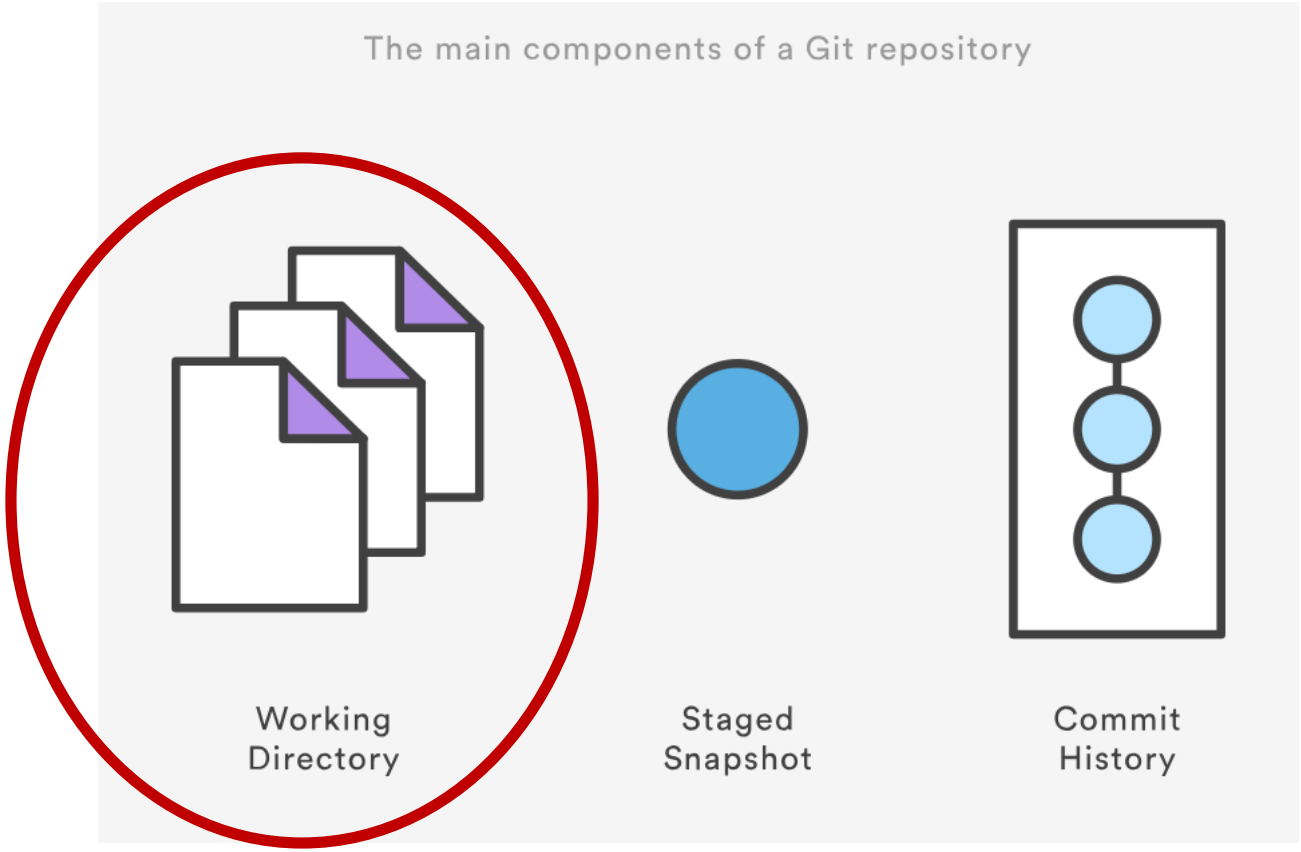
Virtual storage of project's entire history – the gits

repo = project repository

This is CRITICAL INFORMATION (not the trees part, that's dumb)

Tree: the working directory (Editing)

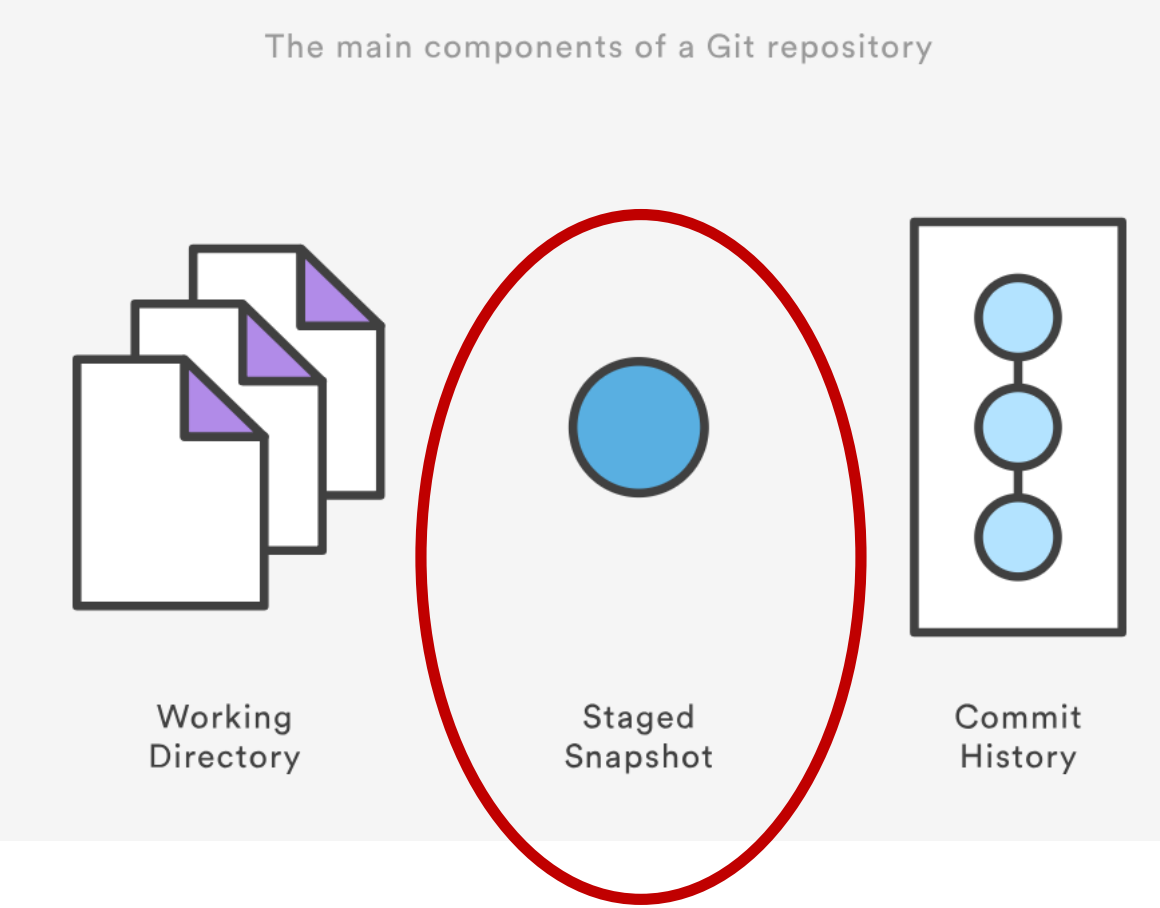
- In sync with the file system
- gedit, Microsoft Word, vim, emacs(but why?)
- Where you work
  - Put your datasets
  - Program your scripts
  - Get your output



Tree 2: The Index (Staging)

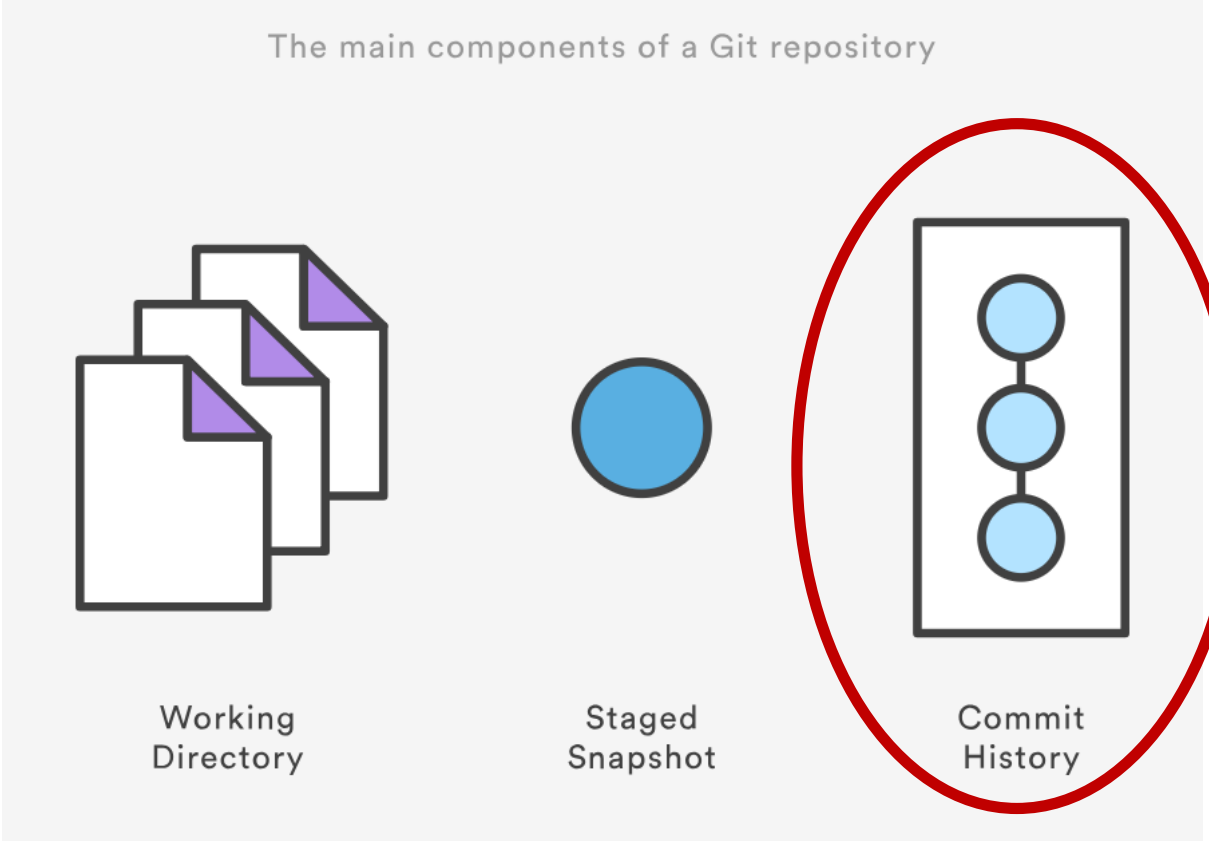
- Stage changes for committing to history
- A buffer between active work and your work's history
- Uses an internal caching mechanism

- The place where you select what goes into a commit



Tree 3: History(Committing)

- Directed acyclic graph (DAG)
- Full commit history
- Permanent (mostly) snapshots of staged changes every time you commit them
- Combination of delta encoding and compression



The Basic Git Workflow (Being Tarzan)

- **Edit** – edit files in your working directory
- **Stage** – add those changes to the staging index, putting together your next commit
- **Commit** – take a snapshot of your collected changes (stage) and add it to your project's official history

Getting starting

git init – create a new repo

- Can run inside empty directory
- Or create a new directory
  - git init <new_directory>
- Or inside a project you've already started
  - ***Nothing you've done is changed, you just can start being cool now**

The .git directory is created inside the folder which contains all the necessary metadata used by git. That's all it does when you run git init.

There's no admin privileges nonsense, it's just creating a new set of directories and files inside your project.

Running git init again won't do anything (used for moving .git and getting new templates, but we don't cover that)

```
149 thallium:/dors/capra_lab/users/sliwosgr% mkdir giterdone
150 thallium:/dors/capra_lab/users/sliwosgr% cd giterdone/
151 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git init
Initialized empty Git repository in /dors/capra_lab/users/sliwosgr/giterdone/.git/
152 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
153 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cd .git
154 thallium:/dors/capra_lab/users/sliwosgr/giterdone/.git% ls
branches/  config  description  HEAD  hooks/  info/  objects/  refs/
155 thallium:/dors/capra_lab/users/sliwosgr/giterdone/.git%
```

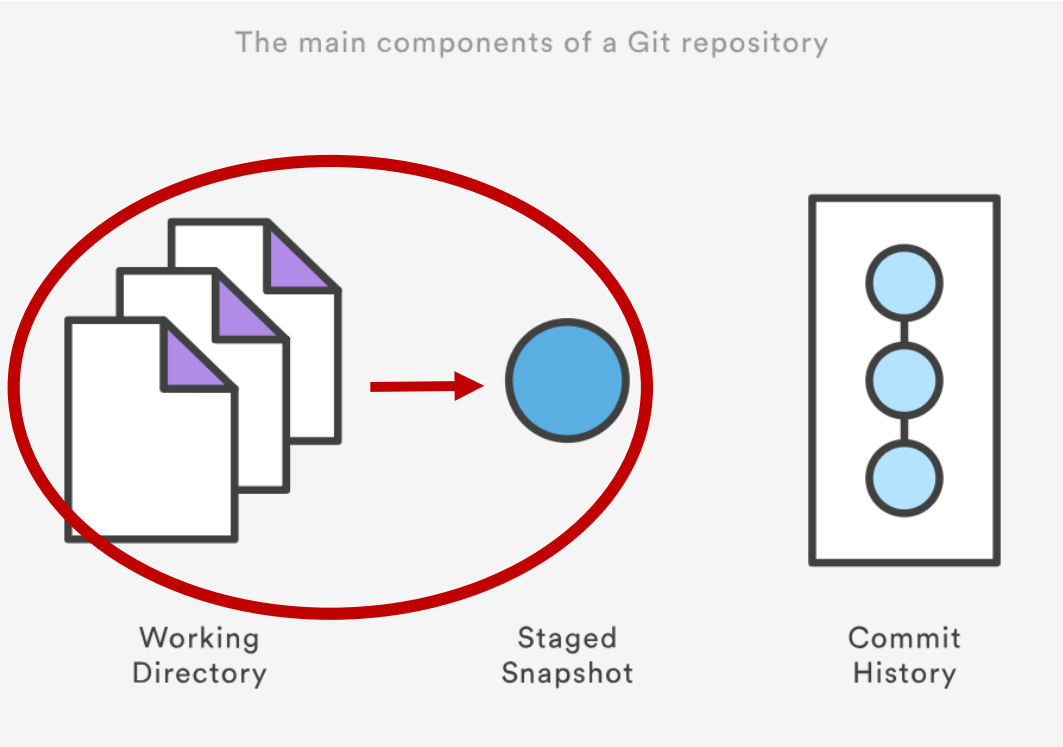
A Git iteration

- 1. Edit: If you don’t know how to edit a file then go home
- 2. Stage
- 3. Commit

2. Stage

git add <filename>

- Adds a change in the working directory to the staging area
- Must specify something to stage
  - . = Add everything
    - useful if you git init in a project already started
  - <dir> = Add a directory
  - -p = Interactive, select which chunks of changes to add
- Doesn't alter your working directory or record anything in history
- Must be repeated every time you edit



Check the state of the working dir and stash

git status

- Shows which changes have been staged and which haven’t
- Lists untracked files: files that have never been staged or committed
- Does not show any history

- Good practice to check before running commit
  - Commits are snapshots in time
  - ‘Atomize’ your progress. Commit units of change.

Stage the first commit

```
176 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "import computerprog
ramming\ngood graphics true\n good sound true\nbad program false\n" > greghub.cp
lusplus
177 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       greghub.cplusplus
nothing added to commit but untracked files present (use "git add" to track)
178 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add greghub.cplusplus

179 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   greghub.cplusplus
#
180 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

Write a program

Nothing staged yet

Stage the new program

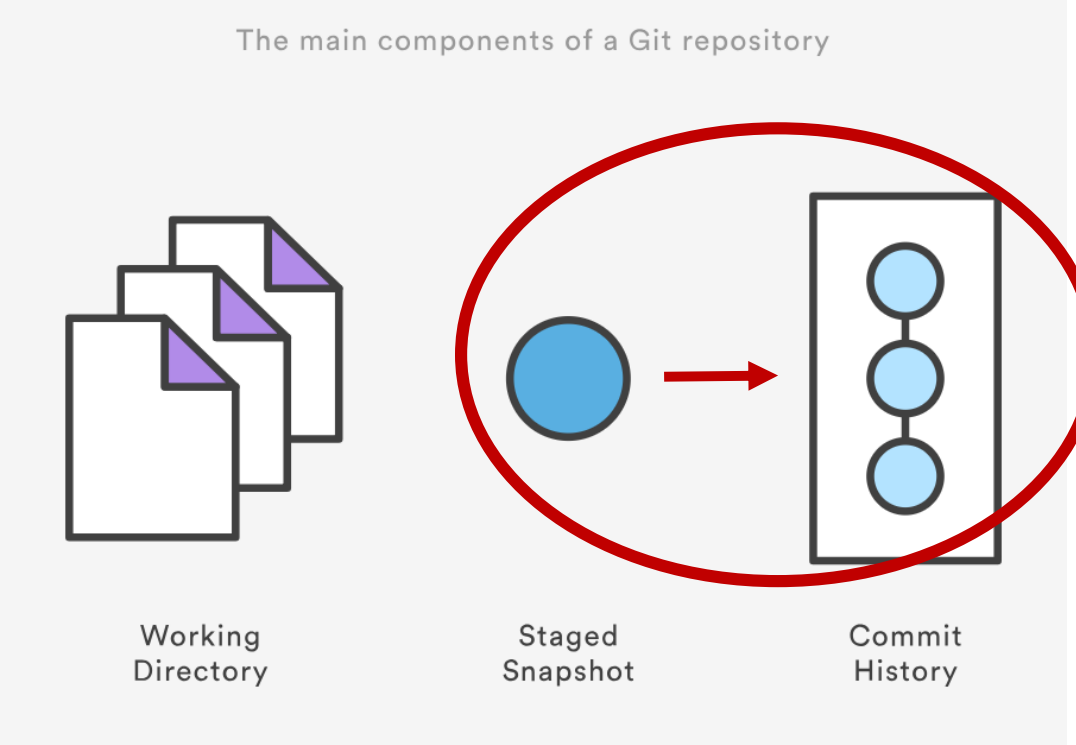
Wow there it is go figure

3. Commit

git commit

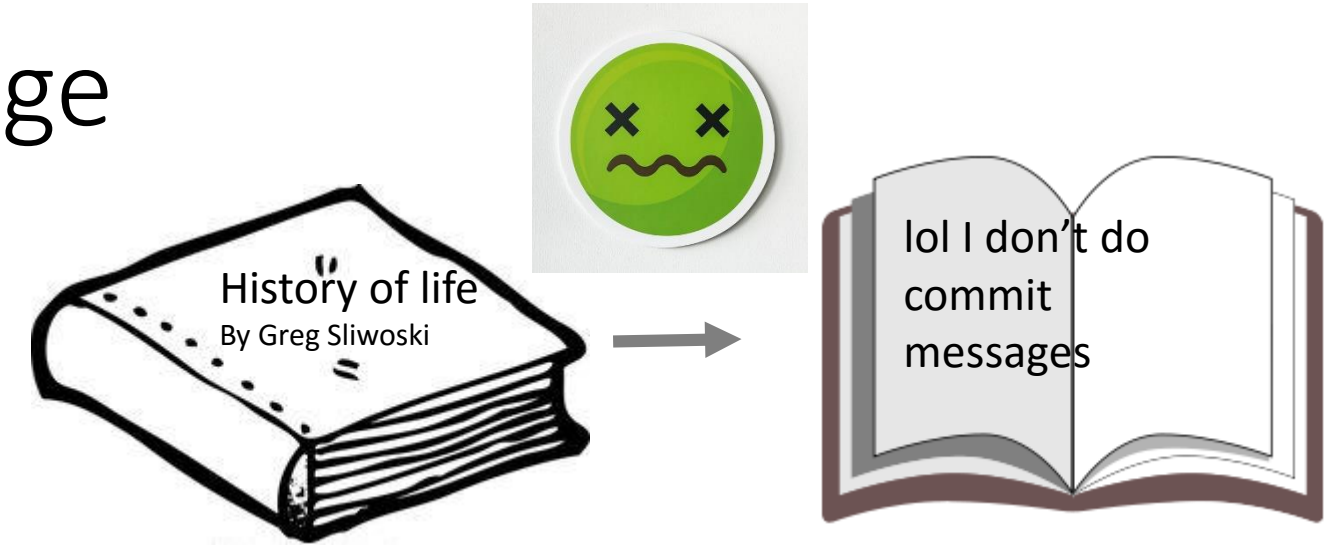
- Commit the current stage to history
  - Hitting ‘save’
  - Permanent unless you explicitly tell git to change your history
    - still permanent but in a secret place
  - Refreshes staging index
  - Doesn’t change anything in working directory
  - Asks for a commit message

- Essentially advances the current branch one snapshot.
  - Each commit is a core building block of project history



Note: if working on a public repo (from Github) this doesn’t change the Github version, just your local one

The commit message



- Critical
- Critical
- Critical
- Common practice:
  - Line 1 = short summary (50 char or less)
  - Line 2 = blank
  - Line 3+ = All changes detailed

- git commit –m "shortcut to supply message to command line"
  - Otherwise git will open default text editor (can be changed)

Let’s commit this sucker

```
185 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git commit -m "Yo whatup
I'm gittinerdone\n\nI basically programmed my own version of git. This is the fi
rst part. It's well written."
[master (root-commit) 960d173] Yo whatup I'm gittinerdone\n\nI basically program
med my own version of git. This is the first part. It's well written.
 1 file changed, 5 insertions(+)
 create mode 100644 greghub.cplusplus
186 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

Let’s add some more stuff to my program

```
105 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
greghub.cplusplus
106 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "This is a new class
\nIt adds AI:\nnew main function\nAIAIAIAIA prediction\nAll done" > AIclass.cplu
splus
107 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "\nAdd the AIclass f
ile I made now greghub writes its own programs nice one greg" >> greghub.cpluspl
us
108 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add .
109 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   AIclass.cplusplus
#       modified:   greghub.cplusplus
#
110 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git commit -m "Added abil
ity of greghub to write its own programs\n\nlol no details oh well"
[master 0221877] Added ability of greghub to write its own programs\n\nlol no de
tails oh well
 2 files changed, 7 insertions(+)
 create mode 100644 AIclass.cplusplus
111 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

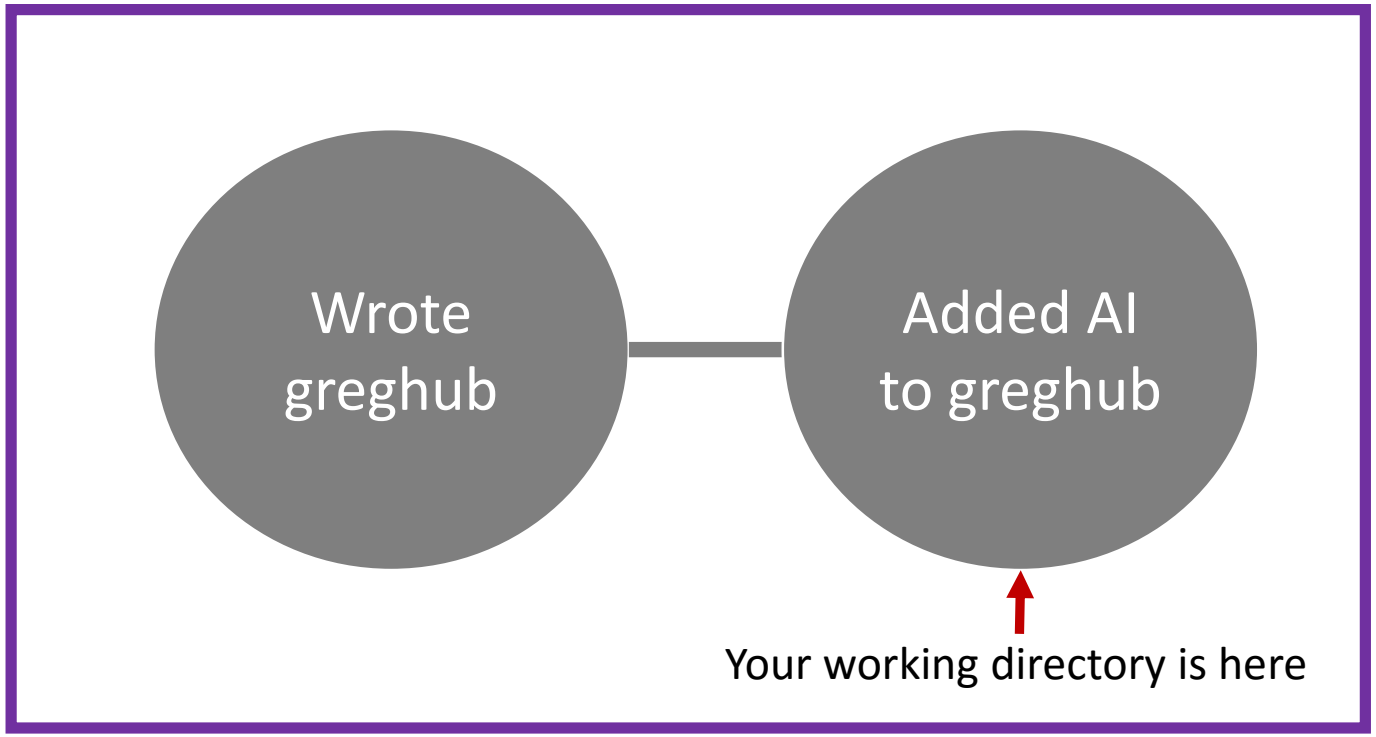
Add new class for AI

Stage it

Commit AI

You can now put git on your resume

And technically be correct 



The giterdone repo

But let’s learn how to use git for real

Not just a record keeper: Git Branches

- Branches are natural (look at trees they branch)
- The git branch creed:

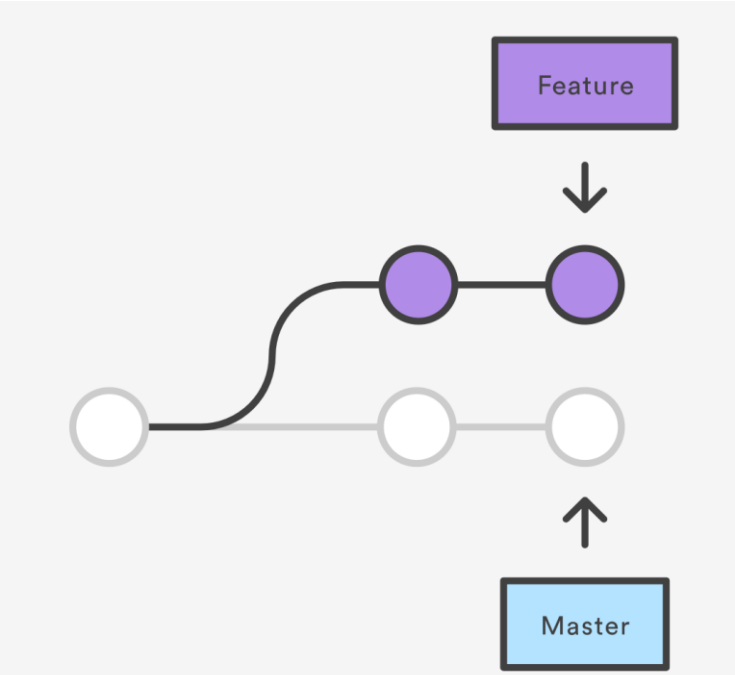
  thy new task

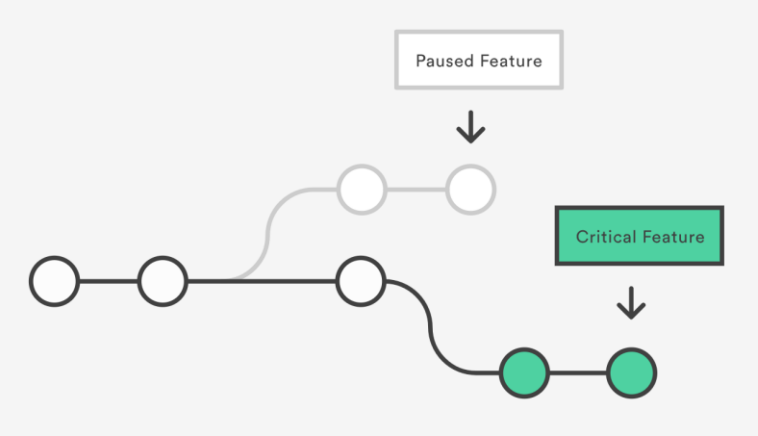
  whether tis Jupiter or Mercury

  you start a new damn branch

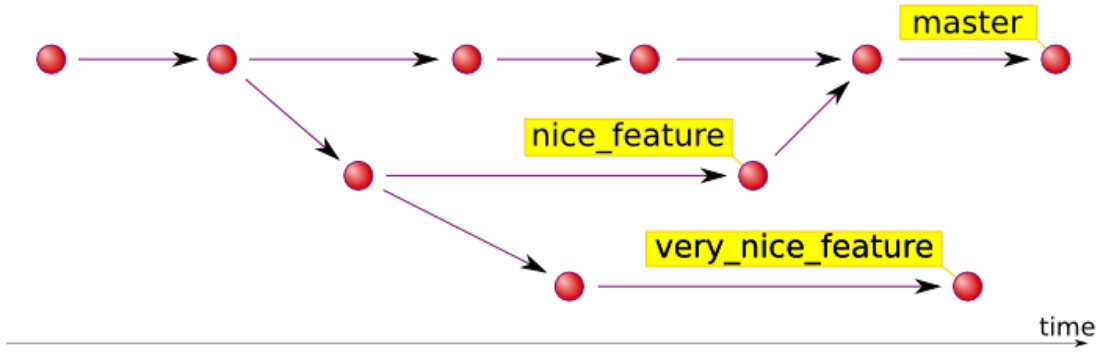
- Branches are isolated working environments
  - Flexibility – no more commenting out unfinished code
  - Play around – whatever you do on a branch doesn't change master (commit nonsense for all I care)

- The **master** branch is production quality at all times
  - Where you started
  - You were already using branches and didn't even know it





Branching

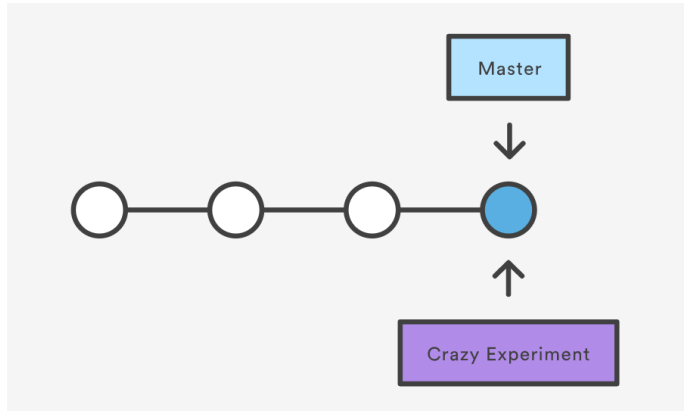


git branch <new_feature>

- Creates a new branch but doesn't move to it (makes no changes)
- git branch = list all branches
- -d = delete (will not delete unmerged changes, -D to force delete)
- -m <new_name> = renames current branch

- You are always on a branch
  - git status shows which one
  - Every time you commit it adds to the branch you are on
- Branches are lightweight
  - Creating a branch simply creates a new pointer to your current commit
    - Nothing in repo changes
  - A branch is just the tip of a set of commits

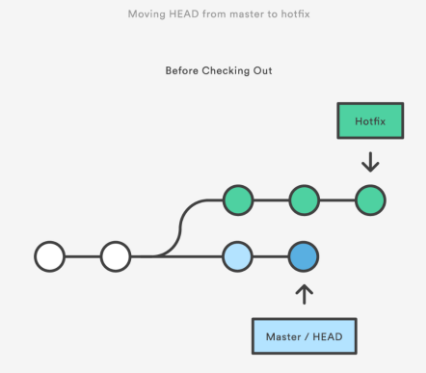
```
109 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
```



Branch jumping

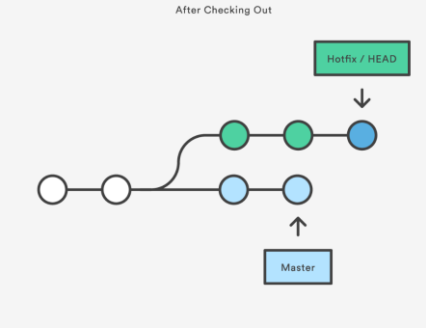
- If you're on master, and you create a new branch, you do not move to it, any further commits attach to master.

git checkout new_feature = go to the new branch



git checkout Hotfix

- Alternatively, create a new branch and move right to it
  - **git checkout –b feature2**
    - Then you can switch back to master: git checkout master
    - And delete the branch: git branch –d feature2



Let’s improve greghub the right way

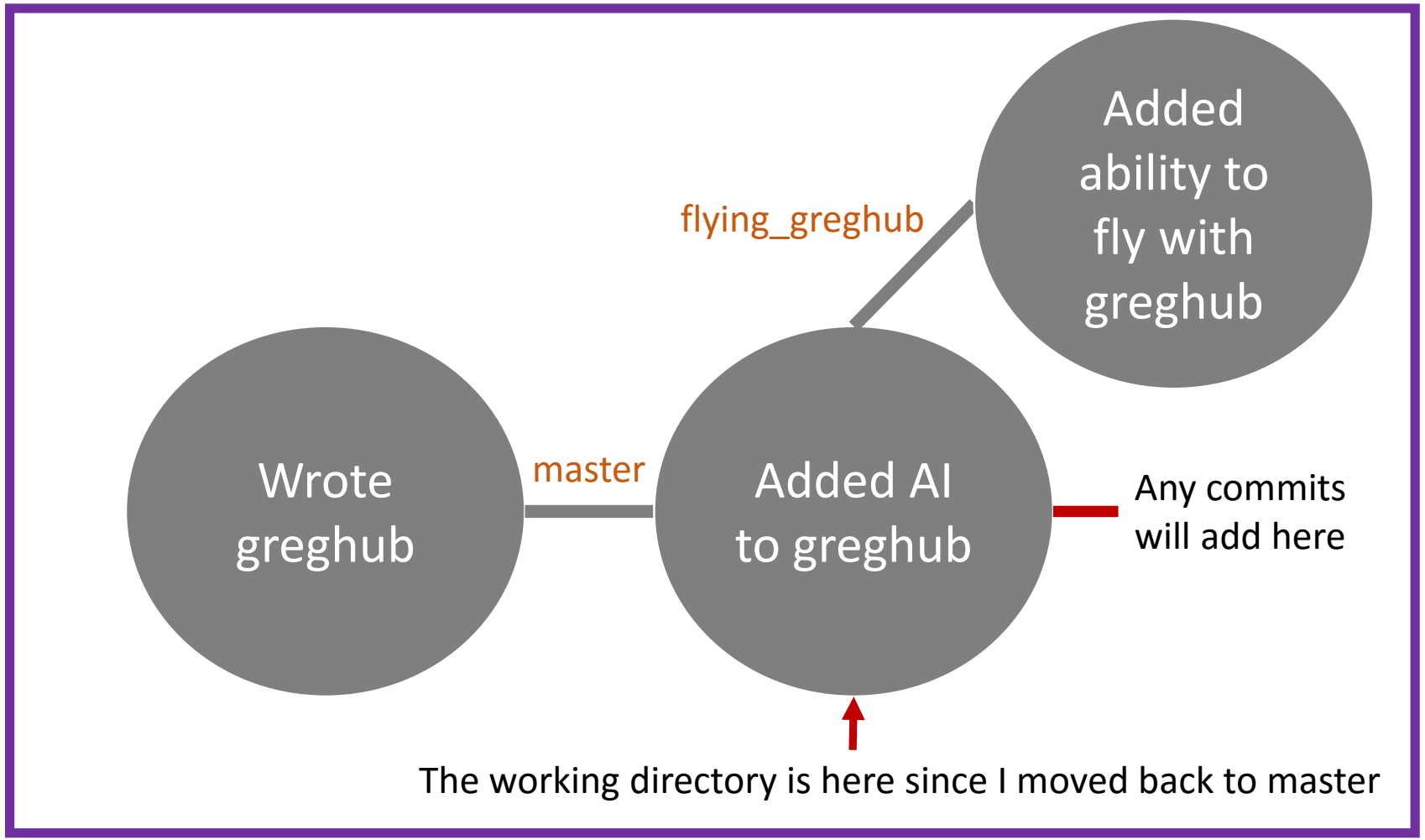
```
112 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout -b flying_gr
eghub
Switched to a new branch 'flying_greghub'
113 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "Include flight\nfly
 function: transform computer into airplane\nadd airplane controls\n" > flying_g
reghub.cplusplus
114 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "\nAdd flying abilit
y function file\n" >> greghub.cplusplus
115 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add .
116 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch flying_greghub
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   flying_greghub.cplusplus
#       modified:   greghub.cplusplus
#
117 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git commit -m "Added abil
ity to fly like an airplane in real life while using greghub\n\nNo details neces
sary"
[flying_greghub 39a769f] Added ability to fly like an airplane in real life whil
e using greghub\n\nNo details necessary
 2 files changed, 7 insertions(+)
 create mode 100644 flying_greghub.cplusplus
118 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout master
Switched to branch 'master'
119 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
AIclass.cplusplus  greghub.cplusplus
120 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

See no flying_greghub on master

Create a new branch for the flight feature instead of just adding to master (in case I mess up which is unlikely but people are already using greghub so I gotta do this right plus I should warn people before they start flying)

I added flight to this branch, but nothing on master has changed!

giterdone is a legit git repo now

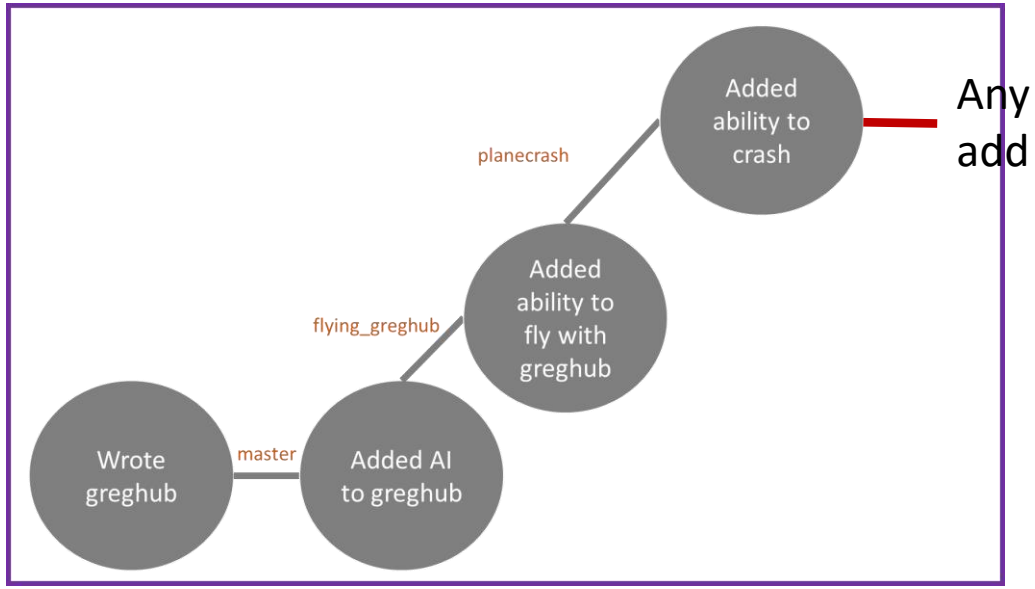


The giterdone repo

Add to any branch from anywhere

git checkout –b <new_branchname> <source_branch>

```
121 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
nothing to commit, working directory clean
122 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout -b planecras
h flying_greghub
Switched to a new branch 'planecrash'
123 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "plane crashes funct
ion\n laugh track\n exit with zero status" > planecrash.cplusplus
124 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "\nadd plane crash f
unction file\n" >> flying_greghub.cplusplus
125 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add .
126 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch planecrash
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   flying_greghub.cplusplus
#       new file:   planecrash.cplusplus
#
127 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git commit -m "Lol I adde
d the ability to crash the plane!\n\nlololololol"
[planecrash 5343311] Lol I added the ability to crash the plane!\n\nlololololol
 2 files changed, 6 insertions(+)
 create mode 100644 planecrash.cplusplus
128 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```



Any commits add here

The Art of the Move

git checkout – moving around a git repo (not just branches)

- Instantly change your entire workspace (*or individual file*) to reflect any state in time or branch
- Since this can lose local changes, git forces you to stash or commit changes in working directory that would be lost
- Not to be confused with git clone (fetches a remote repo and puts you in it)

Internally, all git checkout does is:

1. Alter HEAD to point to your new location
   - what is HEAD?
2. Change your working directory to match that new location

- Specifying a branch name moves HEAD to the last commit of that branch
- Specifying a file replaces just that file in your working directory

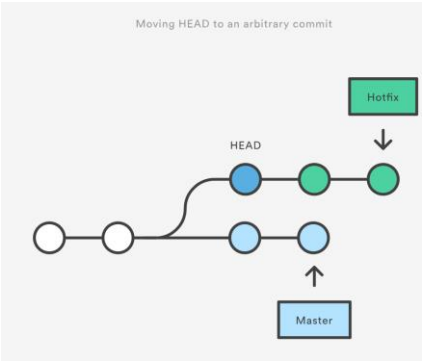
- You can move to any specified commit but it results in a detached head state
  - What's a detached head state and how do you specify commits directly?

Individual commits: ID or ref

- All commits have unique SHA1 ID (reported after commit)

```
[master (root-commit) 960d173]
```

- **refs** are convenience aliases for commits
  - Branch names are special refs to the last commit in that branch
  - HEAD is just a special ref
  - You don't need to specify the full ref name
- relative refs: use ~n (ex: HEAD~2 = grandparent of HEAD)



- refs are Stored as normal text files in .git/refs
- .git/refs/heads/ = all local branches in repo
  - each filename is a branch and inside is the commit hash at the tip of that branch
- To change location of master branch, git just changes contents of refs/heads/master file
- Creating a new branch just writes a commit hash to a new file
- Garbage collection routinely compresses refs into a single file for efficiency.
  - if refs dir is empty that's where they went

```
135 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cd .git/refs/heads/
136 thallium:/dors/capra_lab/users/sliwosgr/giterdone/.git/refs/heads% ls
flying_greghub  master  planecrash
137 thallium:/dors/capra_lab/users/sliwosgr/giterdone/.git/refs/heads% cat plane
crash
5343311613a0a94ff4b6b994a32e74aeaa8a7c4d
```

Tags

git tag <tagname>

- ref that points to specific points in git history
  - Represent read-only branch
- Typically used to mark version release
- By default tags whatever HEAD currently points to
- git tag = list tags
- git tag –d <tagname> = delete a tag
- git checkout <tagname> = go to a tag (causes detached HEAD state)

- Types of tags
  - Lightweight: bookmarks to a commit (usually private)
  - Annotated: stores meta data such as tagger name, date, a message (usually public)
    - git tag –a v1.4 –m “Version 1.4”

Special refs

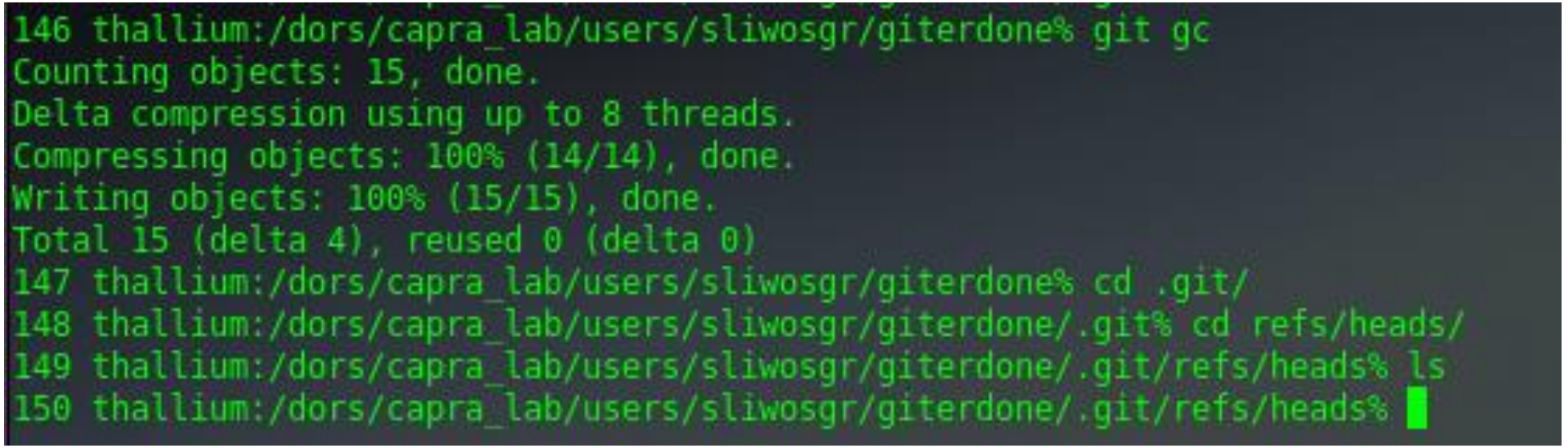
- Found in top level .git directory
- **HEAD = currently checked-out commit/branch**
  - For the most part, HEAD is the only reference you'll be using directly

```
138 thallium:/dors/capra_lab/users/sliwosgr/giterdone/.git/refs/heads% cd ../../

139 thallium:/dors/capra_lab/users/sliwosgr/giterdone/.git% cat HEAD
ref: refs/heads/planecrash
140 thallium:/dors/capra_lab/users/sliwosgr/giterdone/.git% git checkout master
fatal: This operation must be run in a work tree
141 thallium:/dors/capra_lab/users/sliwosgr/giterdone/.git% cd ..
142 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout master
Switched to branch 'master'
143 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cd .git/
144 thallium:/dors/capra_lab/users/sliwosgr/giterdone/.git% cat HEAD
ref: refs/heads/master
```

- ORIG_HEAD = backup ref to HEAD before drastic changes to it
- MERGE_HEAD = commit(s) that you're merging into the current branch with git merge.
- FETCH_HEAD = most recently fetched branch
  - running git pull runs git fetch which updates FETCH_HEAD and runs git merge FETCH_HEAD

Garbage collection

- By default runs every 90 days
- Cleans up orphaned commits and compresses stuff
- Can invoke with `git gc`
- Watch my refs vanish (they got combined into one file):



Detached HEAD State

- HEAD should point to a branch name
  - branch name is just a special ref to last commit in branch
  - HEAD can point to any snapshot in history (it's just a file with an ID in it)
- When HEAD doesn't point to a branch ref, you're in a detached HEAD state
  - checkout a tag (tags are read-only branches) (git checkout v1.0)
  - checkout a specific commit ID (git checkout 5343311)
  - checkout a relative (git checkout HEAD~2)
- Any commits to a detached HEAD state are orphaned
  - Have no associated branch
  - Will be removed by garbage collection
  - Creating a branch from a detached HEAD will make it all good!

```
154 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout 5343311
Note: checking out '5343311'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 5343311... Lol I added the ability to crash the plane!\n\nlololo
ol
155 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "Make the laugh tra
k play twice for good measure\n" >> planecrash.cplusplus
156 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add planecrash.cplus
lus
157 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git commit -m "Make the
augh track play twice when you crash\n\nlol*2"
[detached HEAD 5a537a0] Make the laugh track play twice when you crash\n\nlol*2
 1 file changed, 2 insertions(+)
158 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout -b woopsie
Switched to a new branch 'woopsie'
159 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
AIclass.cplusplus         greghub.cplusplus
flying_greghub.cplusplus  planecrash.cplusplus
160 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat planecrash.cplusplus

plane crashes function
 laugh track
 exit with zero status
Make the laugh track play twice for good measure
```

Putting things back together

git merge <source_branch>

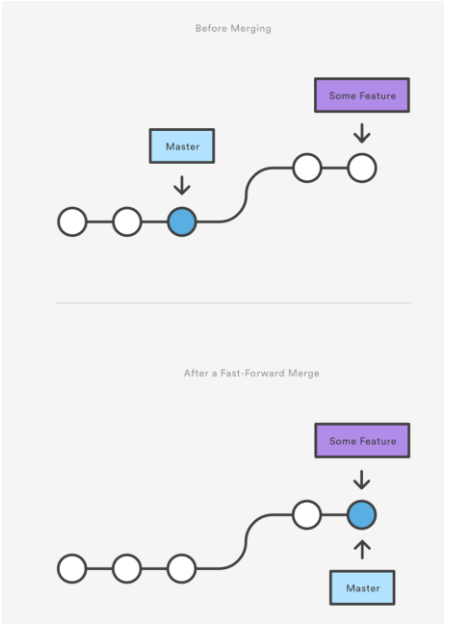
- Add all the commits from one branch to another
  - Combine independent lines of creation
  - Add finished features to master
  - Merges INTO current branch (source_branch is completely unaffected)
    - Merge into a different branch than the current one with git merge <dest> <source>

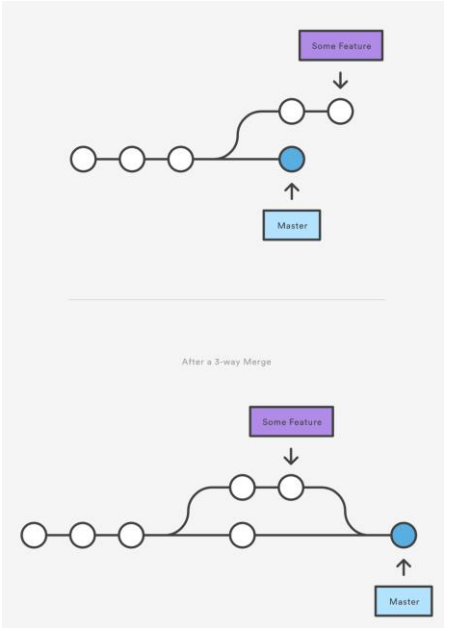
- Typical workflow after finishing feature:
  1. git checkout master
  2. git merge feature_branch
  3. git branch –d feature_branch (delete the old feature_branch since it's now part of master)

Protip: commit any changes in your working dir before you run a merge

2 Types of merge

1. Fast-forward merge (ff)
   - Simple linear merge
   - Does not create new commit (--no-ff to force it to)
   - Essentially just moves HEAD to point to very last commit of other branch and straightens into one branch
   - Looks like you never created a branch in the first place
2. 3-way merge
   - When branches have diverged
   - Generates a merge commit
   - May require conflict resolution





--ff-only = only merge if current HEAD is up to date or merge can resolve as ff.

Bye branches

```
162 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout master
Switched to branch 'master'
163 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git merge woopsie
Updating 0221877..5a537a0
Fast-forward
 flying_greghub.cplusplus | 7 +++++++
 greghub.cplusplus        | 3 +++
 planecrash.cplusplus     | 5 +++++
 3 files changed, 15 insertions(+)
 create mode 100644 flying_greghub.cplusplus
 create mode 100644 planecrash.cplusplus
164 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
nothing to commit, working directory clean
165 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
AIclass.cplusplus         greghub.cplusplus
flying_greghub.cplusplus  planecrash.cplusplus
```

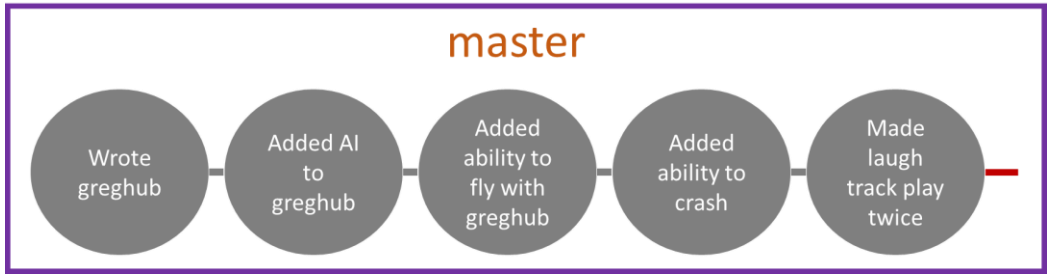
Linearize master

```
170 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git branch
  flying_greghub
* master
  planecrash
  woopsie
171 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git branch -d flying_greg
hub
Deleted branch flying_greghub (was 39a769f).
172 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git brand -d planecrash
git: 'brand' is not a git command. See 'git --help'.

Did you mean this?
        branch
173 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git branch -d planecrash
Deleted branch planecrash (was 5343311).
174 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git branch -d woopsie
Deleted branch woopsie (was 5a537a0).
175 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git branch
* master
176 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

lol yes

Delete my old branches

My repo now looks like this



Conflict resolution

- When merge can’t reconcile changes
  - HEAD stays the same
  - MERGE_HEAD is set to other branch
  - Paths that merged cleanly are updated in index and working tree
  - For conflicting paths, index gets 3 versions (common ancestor, HEAD, and MERGE_HEAD)
  - Working files contain merge result:
    - <<<<<<< yours:sample.txt
    - the merge to part
    - =======
    - the merge from part
    - >>>>>>> theirs:sample.txt

  - git merge --abort = only usable after a failed merge, will reset to pre-merge attempt

Indecisive Greg

```
211 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout -b sorry
Switched to a new branch 'sorry'
212 thallium:/dors/capra_lab/users/sliwosgr/giterdone% sed -ie 's:laugh:cry:g' p
lanecrash.cplusplus
213 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add pla
planecrash.cplusplus   planecrash.cpluspluse
213 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add planecrash.cplusp
lus
214 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git commit -m "I felt bad
 for laughing"
[sorry 8876e08] I felt bad for laughing
 1 file changed, 2 insertions(+), 2 deletions(-)
215 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout master
Switched to branch 'master'
216 thallium:/dors/capra_lab/users/sliwosgr/giterdone% sed -ie 's:laugh:laughha
dhaha:g' planecrash.cplusplus
217 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add plan
planecrash.cplusplus   planecrash.cpluspluse
217 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add planecrash.cplusp
lus
218 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git commit -m "no i didn
t feel sorry"
[master 004b7d8] no i didn't feel sorry
 1 file changed, 2 insertions(+), 2 deletions(-)
```

1. Cry instead of laugh?

2. Hell no, laugh harder

```
219 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git merge sorry
Auto-merging planecrash.cplusplus
CONFLICT (content): Merge conflict in planecrash.cplusplus
Automatic merge failed; fix conflicts and then commit the result.
220 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat pla
planecrash.cplusplus   planecrash.cpluspluse
220 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat planecrash.cplusplus
plane crashes function
<<<<<<< HEAD
 laughhardhaha track
 exit with zero status
Make the laughhardhaha track play twice for good measure
=======
 cry track
 exit with zero status
Make the cry track play twice for good measure
>>>>>>> sorry

221 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git merge --abort
222 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat plan
planecrash.cplusplus   planecrash.cpluspluse
222 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat planecrash.cplusplus
plane crashes function
 laughhardhaha track
 exit with zero status
Make the laughhardhaha track play twice for good measure
```

3. I can’t decide!

4. Oh to hell with it, laugh harder it is

Tips for resolving conflicts

- Use a tool such as git mergetool or a visual tool (kdiff3)
- A common setting is `**git config --global merge.conflictstyle diff3**`
  - This shows the code at their common ancestor when looking at conflicting sections (|||)
- If you have a lot of conflicts in a long piece of content, try git merge --strategy-option=patience
  - http://git.661346.n2.nabble.com/Bram-Cohen-speaks-up-about-patience-diff-td2277041.html
- SourceTree is also a good git gui that can help with conflicts
  - https://www.sourcetreeapp.com/
- Hear from other devs
  - https://developer.atlassian.com/blog/2015/12/tips-tools-to-solve-git-conflicts/

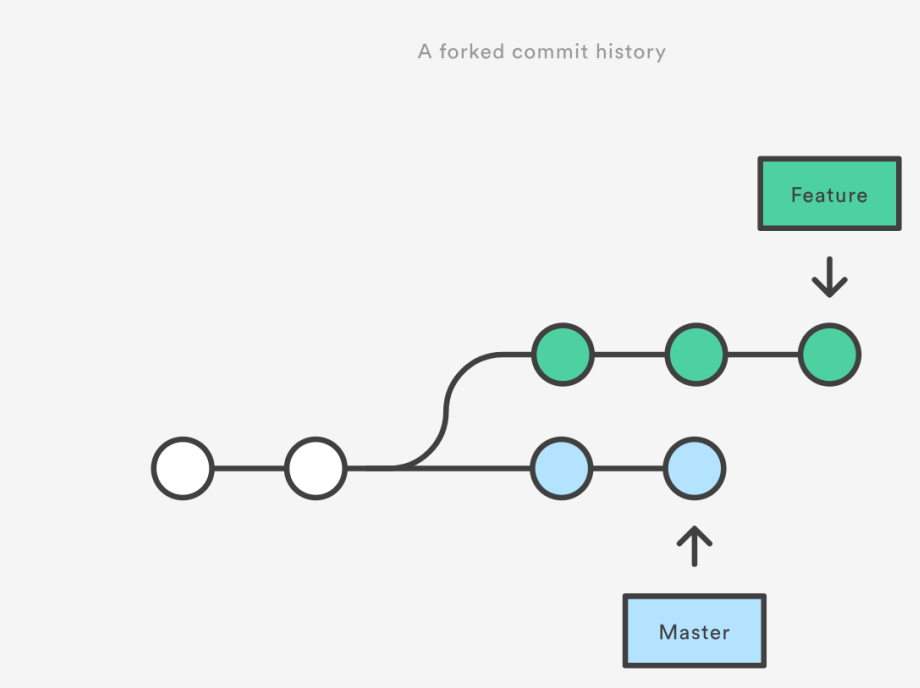
It was like that when I got here: rebasing

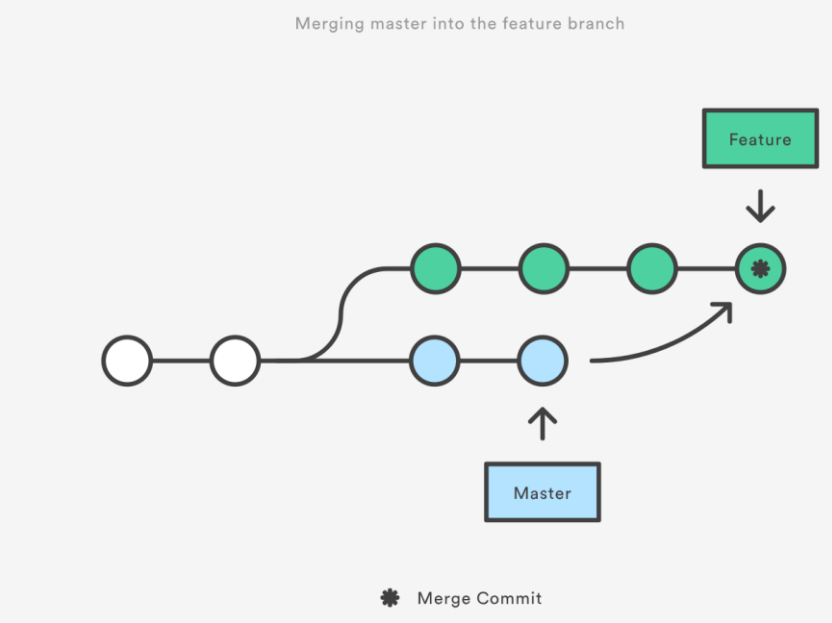
git rebase <source>

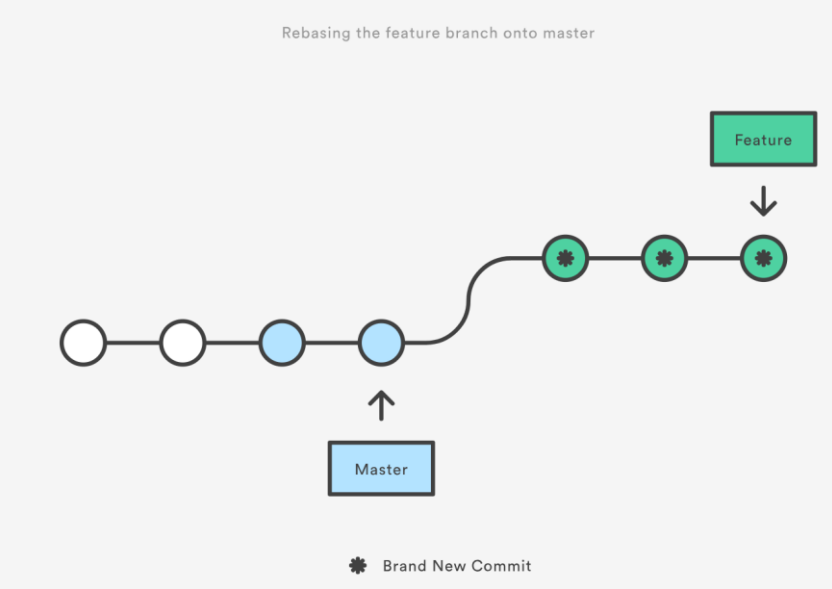
- Alternative to merge with cleaner history
- Common scenario
  - You start working on a new feature
  - Someone updates the master branch
  - You want to merge those into your feature branch (no conflicts)
  - Merging will create a commit
  - You keep merging new changes = more and more commits
- When you rebase *instead of merge*
  - Your feature branch is moved to the tip of master
  - It's as if you started your branch where master is now rather than then
  - You just rewrote history

lol don't do this to public repo's

Rebase vs Merge







How to view your project history

git log

- -n = show n commits
- --oneline = condense each commit to one line
  - you did summarize with 50 char or less right?
- -p show complete diff of each commit (most detailed view possible)
- --grep="<pattern>" = search commit messages (-i to ignore case)
  - You detailed them right?
- <since>..<until>
- --after="2014-7-1" or ="yesterday" or "1 week ago"
- -- <filename> = for a single file
- <branchname> = history of a particular branch

- The pickax: -S"<string>" or –G"<regex>" = search for a particular line of code

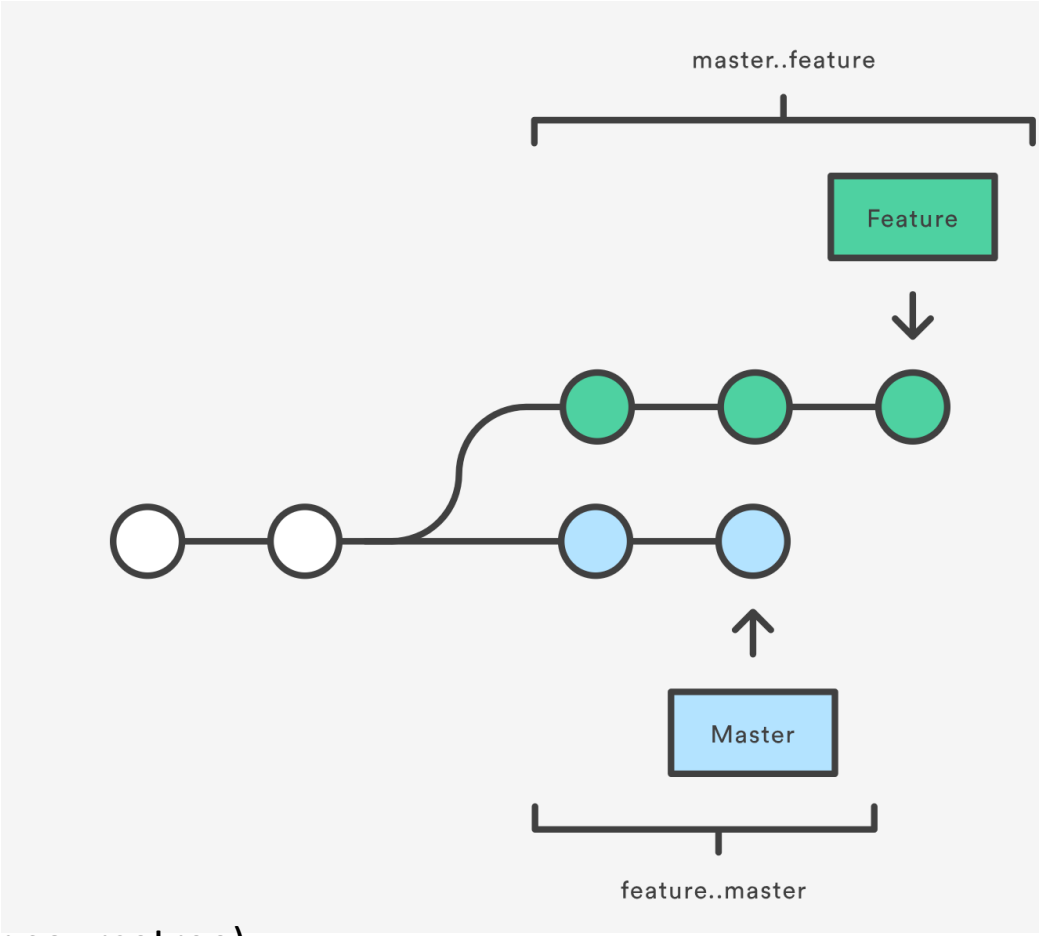
Looking at progress differences between branches

git log master..feature

How far feature has progressed

git log feature..master

How the master has changed since the feature began



For complex histories, there are a ton of visualization tools (gitk or sourcetree)

Comparing snapshots/files/branches

git diff

Compare 2 things:

- Default: any uncommitted changes since last commit
- 2 branchs: git diff branch1..branch2
- 2 Files between branches: git diff branch1 branch2 ./filename.txt
- Working directory against index: git diff ./path/to/file
- 2 refs
  - Default is to compare against HEAD
  - Get refs to compare against using git log

- Often you want to preview changes before merging: git diff <source_branch> <target_branch>

Man my project has really devolved

```
227 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git log
commit 004b7d86c462c803fd260afb7a31f86f128a1f2c
Author: Gregory Sliwoski <gregory.r.sliwoski@vanderbilt.edu>
Date:   Mon May 7 18:52:22 2018 -0500

    no i didn't feel sorry

commit 5a537a0291e9a0b7d9227b429807f4e8ed9485a8
Author: Gregory Sliwoski <gregory.r.sliwoski@vanderbilt.edu>
Date:   Mon May 7 18:24:43 2018 -0500

    Make the laugh track play twice when you crash\n\nlol*2

commit 5343311613a0a94ff4b6b994a32e74aeaa8a7c4d
Author: Gregory Sliwoski <gregory.r.sliwoski@vanderbilt.edu>
Date:   Mon May 7 17:25:45 2018 -0500

    Lol I added the ability to crash the plane!\n\nlololololol

commit 39a769f67788d69cf82704b2be97bc1d9cf38f4c
Author: Gregory Sliwoski <gregory.r.sliwoski@vanderbilt.edu>
Date:   Mon May 7 17:12:46 2018 -0500

    Added ability to fly like an airplane in real life while using greghub\n\nNo

commit 022187748b53fb5ed77d7d6ec33340464a0f5759
Author: Gregory Sliwoski <gregory.r.sliwoski@vanderbilt.edu>
Date:   Mon May 7 16:49:51 2018 -0500

    Added ability of greghub to write its own programs\n\nlol no details oh well

commit 960d173c4e63deb0afd8517eb2ba632680e729a6
Author: Gregory Sliwoski <gregory.r.sliwoski@vanderbilt.edu>
Date:   Fri May 4 16:45:23 2018 -0500

    Yo whatup I'm gittinerdone\n\nI basically programmed my own version of git.
(END)
```

```
245 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git log --decorate --onel
ine --all
ebb93bb (sorry) I felt bad for laughing
004b7d8 (HEAD, master) no i didn't feel sorry
5a537a0 Make the laugh track play twice when you crash\n\nlol*2
5343311 Lol I added the ability to crash the plane!\n\nlololololol
39a769f Added ability to fly like an airplane in real life while using greghub\n
0221877 Added ability of greghub to write its own programs\n\nlol no details oh
960d173 Yo whatup I'm gittinerdone\n\nI basically programmed my own version of g
(END)
```

Where did it all go so wrong?

```
261 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git log --oneline
004b7d8 no i didn't feel sorry
5a537a0 Make the laugh track play twice when you crash\n\nlol*2
5343311 Lol I added the ability to crash the plane!\n\nlololololol
39a769f Added ability to fly like an airplane in real life while using greghub\n
0221877 Added ability of greghub to write its own programs\n\nlol no details oh
960d173 Yo whatup I'm gittinerdone\n\nI basically programmed my own version of g
262 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
262 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
262 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git diff 5343311
diff --git a/planecrash.cplusplus b/planecrash.cplusplus
index bf15f80..2372fa4 100644
--- a/planecrash.cplusplus
+++ b/planecrash.cplusplus
@@ -1,3 +1,5 @@
 plane crashes function
- laugh track
+ laughhardhaha track
   exit with zero status
+Make the laughhardhaha track play twice for good measure
+
263 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

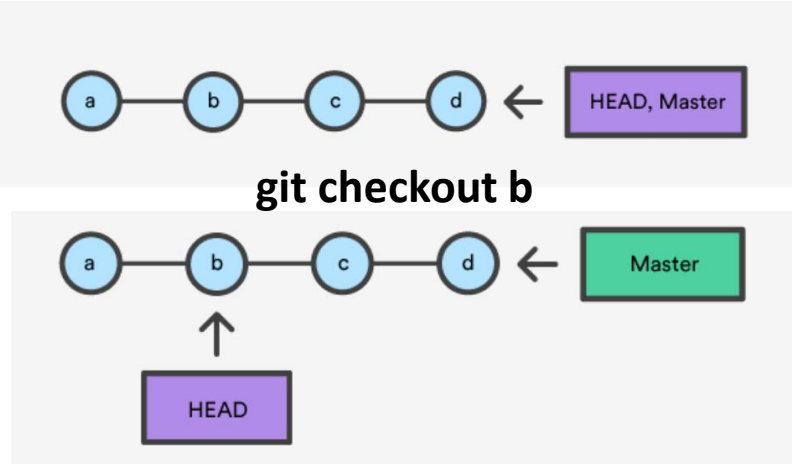
Oh yeah, I added the ability for my users to crash, literally.

Ohhhh, I laughed at them too. Oops.

Okay, this is bad.

Time traveling in git

- Technically there's no such thing as an 'undo' in git.
- Every action is a snapshot of a point in time, even undoing a change
- Simple time travel
  a) Look at an old version
    1. git log to get SHA1
    2. git checkout SHA1 (detached HEAD)
    3. Look around
    4. git checkout master (or whatever branch you were on to go back)
  b) Amend a commit
    - You forgot to add a file or want to change the last commit message
    - **git commit –amend**
  c) Revert a file you changed poorly
    - **git checkout <commit_version> -- <filename>**
      - -- tells git it's a file not a branch, unnecessary if no branch with that name
    - Only changed in working dir: followed by git add <filename>; git commit to keep changes



Undoing commits in git

The bad way:

1. git checkout a previous version
2. Git checkout –b new_branch to move the detached HEAD to a new branch.
3. Go ahead and do your thing and merge it back to master

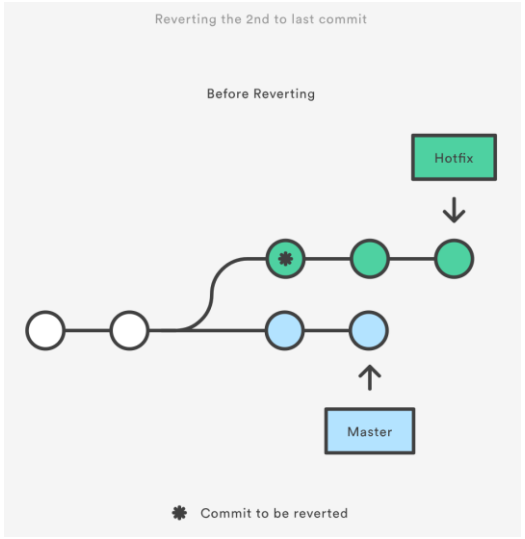
The good way:

a) git revert – public
b) git reset – private

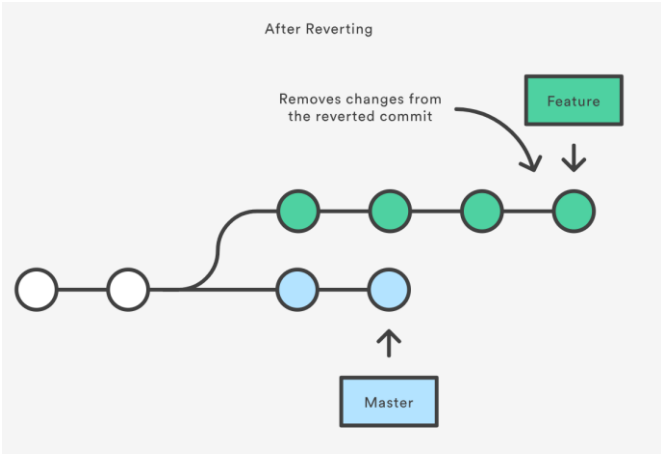
Undoing **public** commits

git revert <commit>

- git revert HEAD = creates new commit that is exact inverse of the last commit
- preserves history
- No file level functionality (only works on commits)
- Only reverts changes in specified commit (not all in between)
  - Specify range with commit1..commit2
- -n just add changes to staging index and working dir, don't commit yet



1. git checkout hotfix
2. git revert HEAD~2



Checkout vs Revert

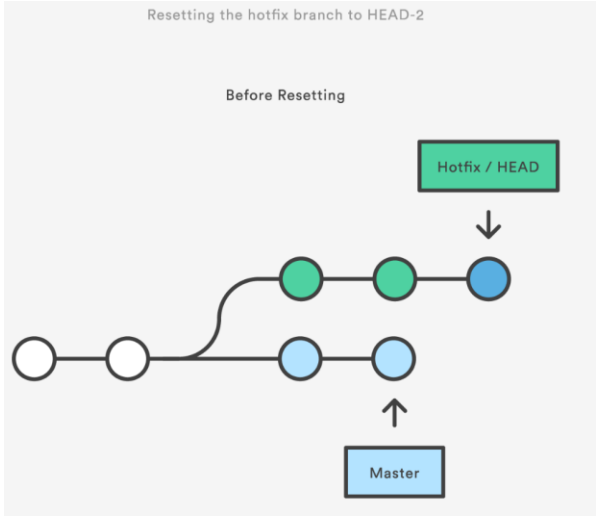
Function	Commit Scope	File Scope
checkout	switch between branches or inspect old snapshots	Discard changes in working directory
revert	Undo commits in a branch	N/A

Undoing **private** changes

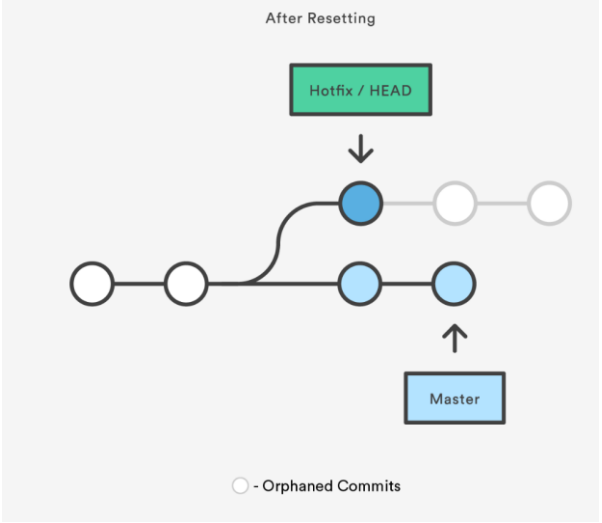
Git reset <commit>

- Commit history is **erased** after the <commit>.
- Will reverse **all** changes after the given commit (unlike revert)
- Those commits are orphaned and will be deleted as garbage

*Git reset <file> just unstages a file (opposite of git add <file>)



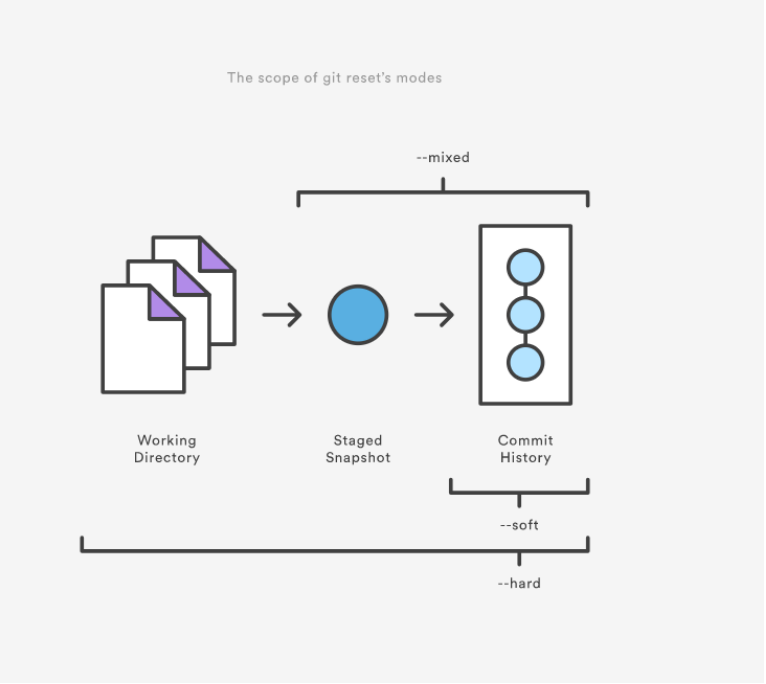
git reset HEAD~2



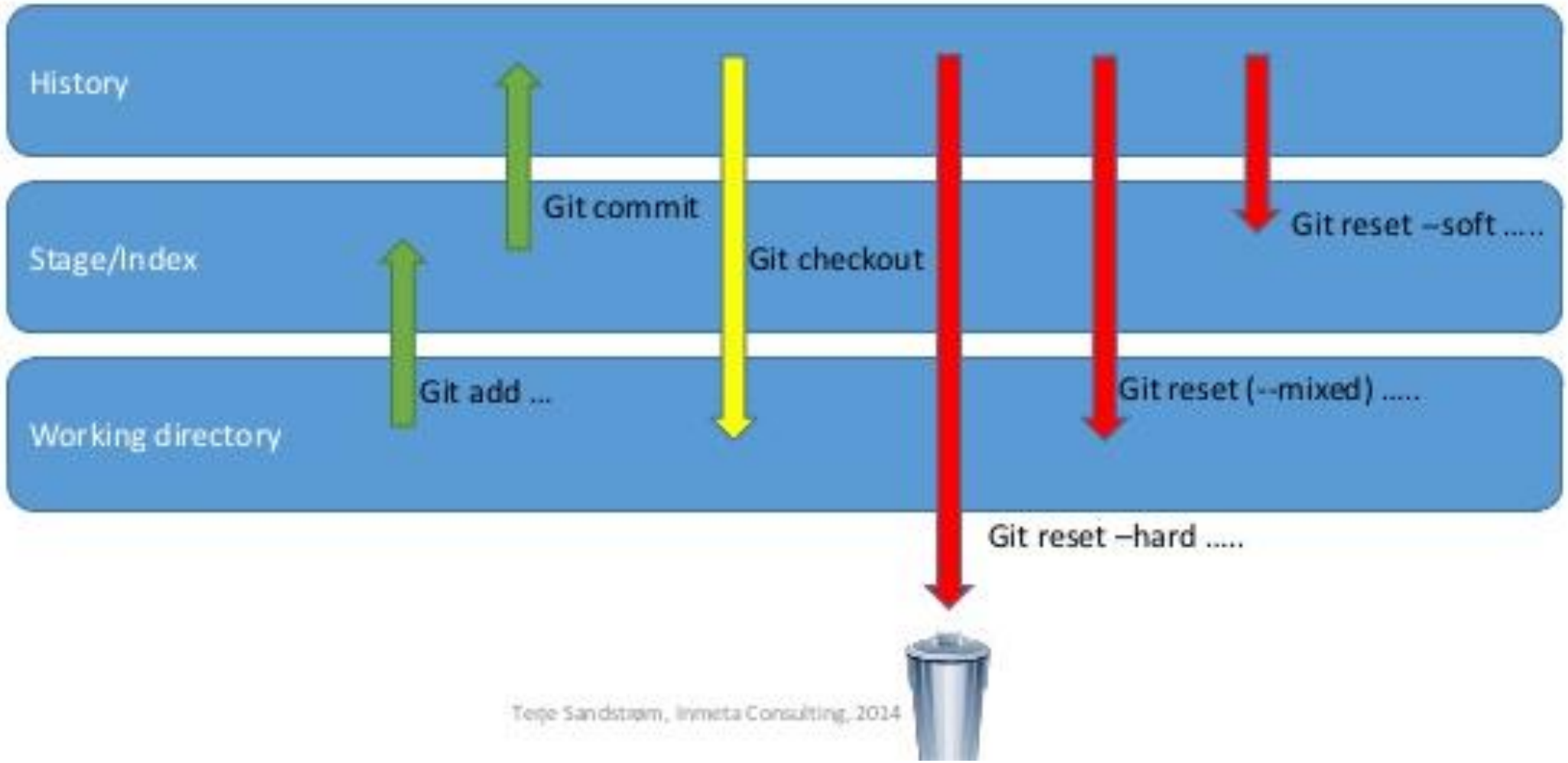
3 Forms of reset

- git reset always moves HEAD, but what else it changes is up to you
  - --soft = commit tree only
  - --mixed = commit tree and staging index [default]
  - --hard = all 3 trees

- Examples:
  - git reset = reset staging area to match most recent commit but leave working dir unchanged.
  - git reset <commit> = move current branch tip backward to commit, reset the staging area to match, but leave working dir alone. All changes made since commit will reside in the working dir.



Git tree movements visualized



Quick fixes to a single file

1. Undo all your edits: **git checkout filename**

```
351 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
nothing to commit, working directory clean
352 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "haha" > flying_greg
hub.cplusplus
353 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
# 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:   flying_greghub.cplusplus
#
no changes added to commit (use "git add" and/or "git commit -a")
354 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout flying_gregh
ub.cplusplus
355 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
nothing to commit, working directory clean
```

Quick fixes to a single file

2. Change it to a previous version: **git checkout commitID filename**

*You can always look at differences first with **git diff ID1..ID2 filename**

```
366 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git log
commit 39a769f67788d69cf82704b2be97bc1d9cf38f4c
Author: Gregory Sliwoski <gregory.r.sliwoski@vanderbilt.edu>
Date:   Mon May 7 17:12:46 2018 -0500

    Added ability to fly like an airplane in real life while using greghub\n\nNo

commit 022187748b53fb5ed77d7d6ec33340464a0f5759
Author: Gregory Sliwoski <gregory.r.sliwoski@vanderbilt.edu>
Date:   Mon May 7 16:49:51 2018 -0500

    Added ability of greghub to write its own programs\n\nlol no details oh well

commit 960d173c4e63deb0afd8517eb2ba632680e729a6
Author: Gregory Sliwoski <gregory.r.sliwoski@vanderbilt.edu>
Date:   Fri May 4 16:45:23 2018 -0500

    Yo whatup I'm gittinerdone\n\nI basically programmed my own version of git.
```

```
367 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git diff 960d17..master g
reghub.cplusplus
diff --git a/greghub.cplusplus b/greghub.cplusplus
index bdbd68c..45f52b8 100644
--- a/greghub.cplusplus
+++ b/greghub.cplusplus
@@ -3,3 +3,8 @@ good graphics true
  good sound true
 bad program false

+
+Add the AIclass file I made now greghub writes its own programs nice one greg
+
+Add flying ability function file
+
368 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout 960d17 gregh
ub.cplusplus
369 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat greghub.cplusplus
import computerprogramming
good graphics true
 good sound true
bad program false

370 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   greghub.cplusplus
#
371 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

Quick fixes to a single file

3. Change it to a version on a different branch: **git checkout branchname filename**

```
356 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git diff master..sorry fl
ying_greghub.cplusplus
diff --git a/flying_greghub.cplusplus b/flying_greghub.cplusplus
index 6d7fa0a..4fe7812 100644
--- a/flying_greghub.cplusplus
+++ b/flying_greghub.cplusplus
@@ -2,3 +2,6 @@ Include flight
 fly function: transform computer into airplane
 add airplane controls

+
+add plane crash function file
+
357 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout sorry flying
_greghub.cplusplus
358 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   flying_greghub.cplusplus
#
```

Quick fixes to a single file

4. Unstage the file: **git reset filename**

5. Go back to the original version (before I started these examples):

```
359 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git reset flying_greghub.
cplusplus
Unstaged changes after reset:
M       flying_greghub.cplusplus
360 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
# 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:   flying_greghub.cplusplus
#
no changes added to commit (use "git add" and/or "git commit -a")
361 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout flying_gregh
ub.cplusplus
362 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
nothing to commit, working directory clean
363 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

Getting back to happier times

```
264 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git log --oneline
004b7d8 no i didn't feel sorry
5a537a0 Make the laugh track play twice when you crash\n\nlol*2
5343311 Lol I added the ability to crash the plane!\n\nlololololol
39a769f Added ability to fly like an airplane in real life while using greghub\n
0221877 Added ability of greghub to write its own programs\n\nlol no details oh
960d173 Yo whatup I'm gittinerdone\n\nI basically programmed my own version of g
265 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
265 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
265 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git reset 39a769f
Unstaged changes after reset:
M       flying_greghub.cplusplus
266 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git reset --hard 39a769f
HEAD is now at 39a769f Added ability to fly like an airplane in real life while
using greghub\n\nNo details necessary
267 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git log --oneline
39a769f Added ability to fly like an airplane in real life while using greghub\n
0221877 Added ability of greghub to write its own programs\n\nlol no details oh
960d173 Yo whatup I'm gittinerdone\n\nI basically programmed my own version of g
268 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

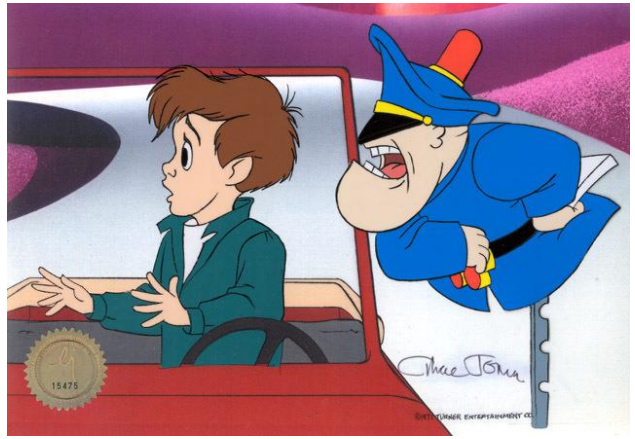
No one will ever know… right?

In case of emergency, break reflog

- **git reflog** = reference logs
  - records every single update to a branch tip, including resets and checkouts
  - Stores everything, committed or not
  - Default is expire after 90 days
  - Stored in dirs: git/logs/refs/heads, .git/logs/HEAD, and .git/logs/refs/stash
  - --all = get a complete reflog
  - show otherbranch = show a specific branch
  - HEAD{<n>} works a lot like HEAD~<n>
- Example:
  - you left some commits orphaned after a git reset.
  - Access them with HEAD@{1}
  - puts you in detached head state so create new branch to continue working

Oh god they know

```
270 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git reflog --all
ebb93bb refs/heads/sorry@{0}: commit: I felt bad for laughing
004b7d8 refs/heads/sorry@{1}: branch: Created from HEAD
39a769f refs/heads/master@{0}: reset: moving to 39a769f
004b7d8 refs/heads/master@{1}: commit: no i didn't feel sorry
5a537a0 refs/heads/master@{2}: reset: moving to HEAD~1
0d01363 refs/heads/master@{3}: merge sorry: Fast-forward
5a537a0 refs/heads/master@{4}: merge woopsie: Fast-forward
0221877 refs/heads/master@{5}: commit: Added ability of greghub to write its own
060d173 refs/heads/master@{6}: commit (initial): Yo whatup I'm gittinerdone\n\nI
```



Some advanced things you might use

- Stashing unfinished changes
  - **git stash**
- Changing default behavior
  - **git config**
- Ignoring unprocessed datasets or other files in your git repo
  - **git ignore**
- Figure out if it was Souhrid who broke the pipeline
  - **git blame**

Stashing changes

Scenario

1. Working on feature
2. There's a bug in the main program
3. Changes not ready for commit but need to switch to master real quick
4. Stash the changes temporarily so you can fix the bug and come back

git stash – save changes (working dir and stage) them for later use.*

git stash pop – reapply staged changes and remove stash

git stash apply – same but keep stash (if applying to multiple branches)

-p = interactive mode, select which chunks to stash

git stash drop = delete stash

*By default git will not stash untracked (new) or ignored files (use –u to stash untracked files, -a include ignored)

```
376 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "import the ability
to control the world\ncontrol the world\ndone" > world_dominashun.cplusplus
377 thallium:/dors/capra_lab/users/sliwosgr/giterdone% echo "add world_dominashu
n ability class" >> greghub.cplusplus
378 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   greghub.cplusplus
#
# 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:   greghub.cplusplus
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       world_dominashun.cplusplus
379 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git stash -u
Saved working directory and index state WIP on master: 39a769f Added ability to
fly like an airplane in real life while using greghub\n\nNo details necessary
HEAD is now at 39a769f Added ability to fly like an airplane in real life while
using greghub\n\nNo details necessary
380 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
AIclass.cplusplus  flying_greghub.cplusplus  greghub.cplusplus
381 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch master
nothing to commit, working directory clean
382 thallium:/dors/capra_lab/users/sliwosgr/giterdone% I fixed a bug here
I: Command not found.
383 thallium:/dors/capra_lab/users/sliwosgr/giterdone% whatevs
whatevs: Command not found.
384 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git stash pop
# On branch master
# 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:   greghub.cplusplus
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       world_dominashun.cplusplus
no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (29980b3f94475ba3e0c734efceac1f0da2962bf4)
385 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
AIclass.cplusplus         greghub.cplusplus
flying_greghub.cplusplus  world_dominashun.cplusplus
386 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

I'm working on a new class

Something's broken so stash my progress

Back to before my changes

Fix my bugs, add, commit, etc.

Bring back my stash

Stashes continued

- Multiple stashes are possible
  - run git stash several times
  - **git stash list** to view them
  - **git stash save “message”** = provide context to a stash
  - **git stash pop stash@{<n>}** = select a specific stash
  - **git stash clear** = delete all stashes

Encoded in repo as commit objects
.git/refs/stash points to most recently created stash
Stash has its own reflog to refer to previous stashes
Since stashes are essentially commits to, you can inspect them with git log
Running git stash creates 2 or 3 new commits:

a new commit to store tracked files in working copy
a pre-existing commit that was at HEAD when you ran stash
a new commit representing the index when you ran stash
(if you used –u or –a) a new commit of untracked files

Configuring git

- All options are held in a 3 plaintext files
  - <reponame>/.git/config = repo specific --local
  - ~/.gitconfig = global settings for all local git repos --global
  - /etc/gitconfig = system wide settings (admin) --system
- You can directly edit config files
- **git config** = convenience command for changing config files
  - Commonly used to define aliases for commands
    - git config –global alias.ci commit = creates an alias for commit so “git ci” can be used instead

```
394 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
395 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git config --local alias
ci commit
396 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[alias]
        ci = commit
397 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat ~/.gitconfig
[credential]
        helper = cache --timeout=3600
[core]
        autocrlf = input
[user]
        name = Gregory Sliwoski
        email = gregory.r.sliwoski@vanderbilt.edu
398 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

Ignore files

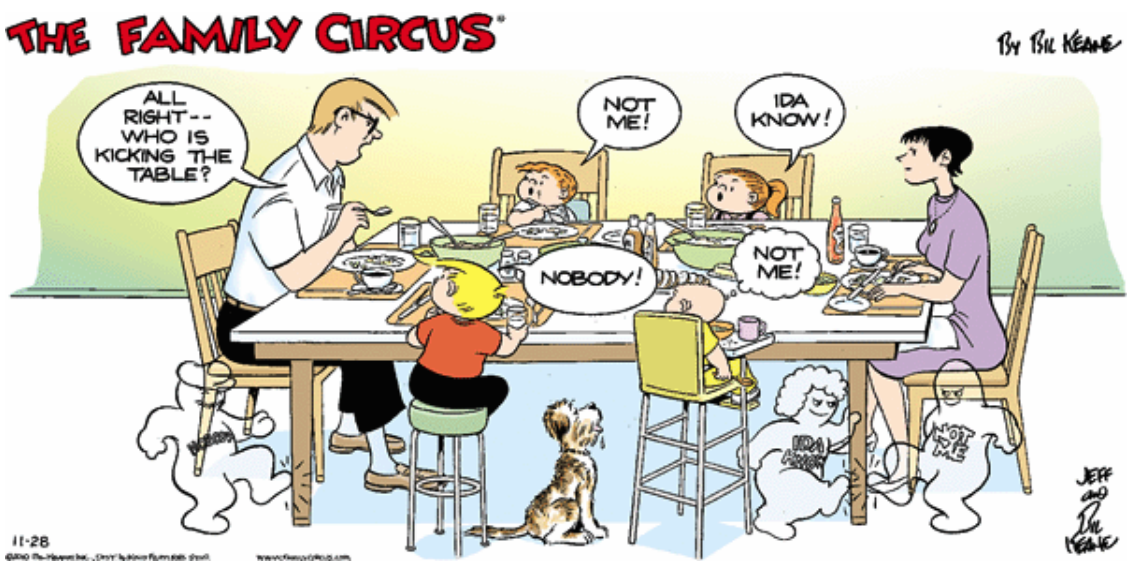
3 File States:

1. Tracked – previously staged or committed
2. Untracked – never staged or committed (new files)
3. Ignored – ignored by git

.gitignore = contains a list of ignored files. Must add to it manually.

- Can use globbing (*,?, [0-9], etc)
- Include comments in ignore with #
- Prepend pattern with ** to match directories

Force git add to include an ignored file with –f <ignored_file>



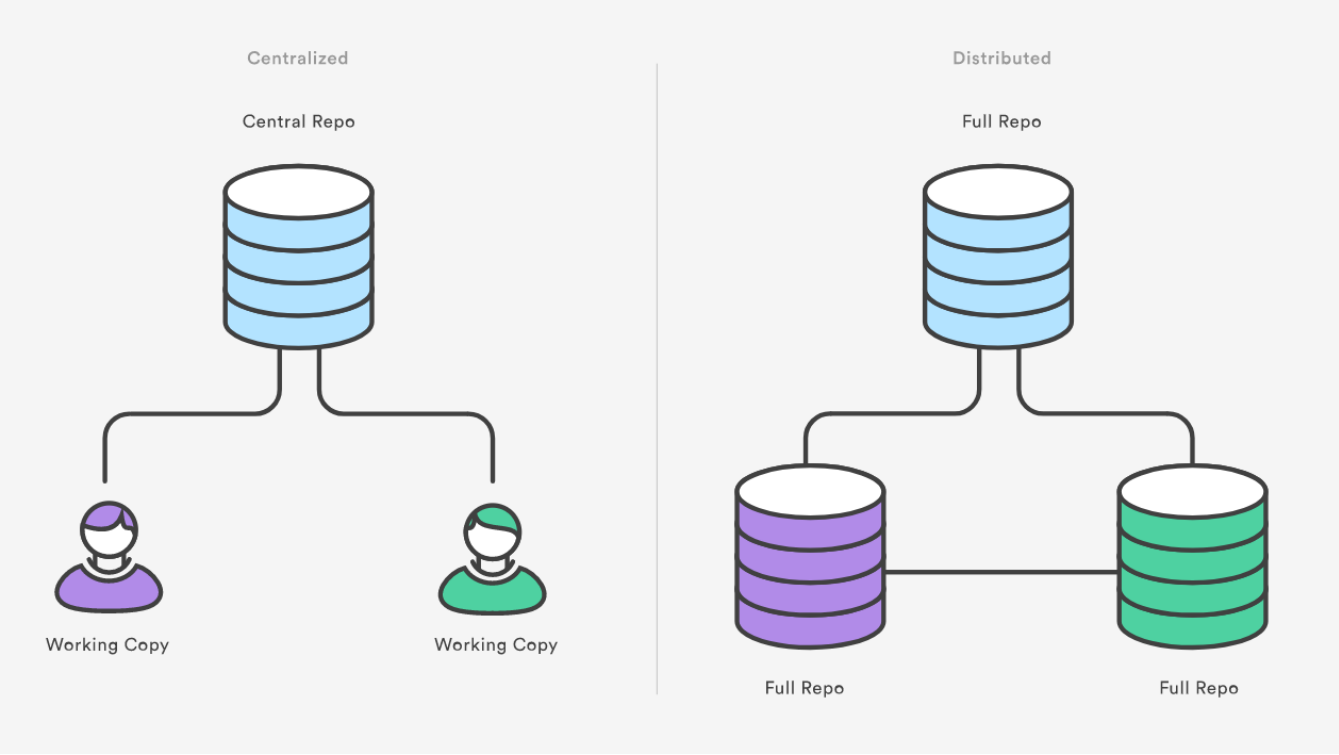
git blame

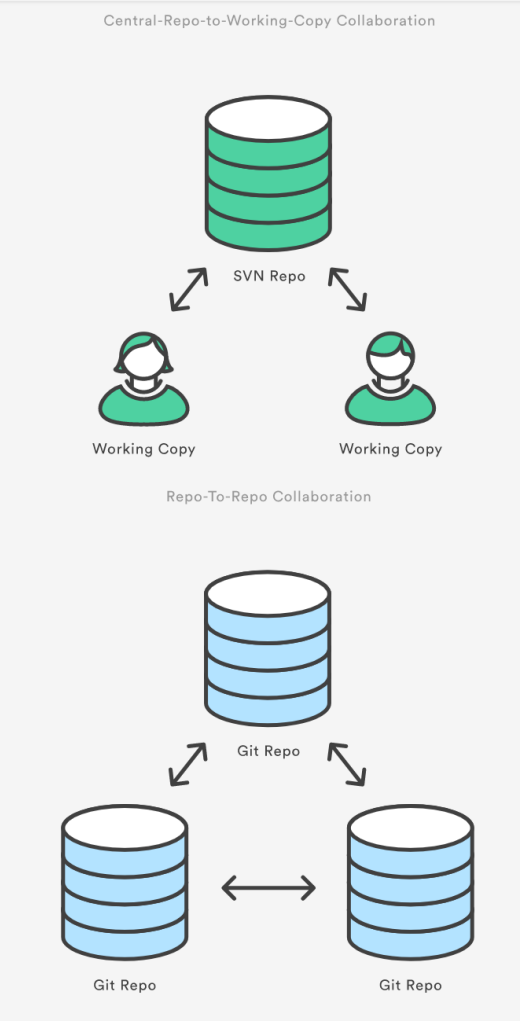
- Display author metadata attached to specific committed lines in a file
- Often used with a GUI display and hosting sites (Github) typically have pretty wrappers for this.
- operates on individual files only.
- Columns are id, author, timestamp, line number, line content
- Ignore whitespace changes with –w
- Get the original author of a line that has been moved (instead of seeing the person who moved it): -M from same file, -C from other file
- Git blame is typically used to see who last edited a line. Further history is more convenient with git log. (git log –S “show me where this code was added or modified all through history”)

```
401 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git blame greghub.cpluspl
us
^960d173 (Gregory Sliwoski  2018-05-04 16:45:23 -0500 1) import computerprogramm
^960d173 (Gregory Sliwoski  2018-05-04 16:45:23 -0500 2) good graphics true
^960d173 (Gregory Sliwoski  2018-05-04 16:45:23 -0500 3)  good sound true
^960d173 (Gregory Sliwoski  2018-05-04 16:45:23 -0500 4) bad program false
^960d173 (Gregory Sliwoski  2018-05-04 16:45:23 -0500 5)
00000000 (Not Committed Yet 2018-05-17 14:40:45 -0500 6) add world_dominashun ab
402 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

Collaborating with git

- Git is **distributed** (unlike SVN which is centralized)
- Git makes no distinction between original and cloned repo's





Getting started

git clone <url>

Copies a remote repo onto local machine (completely isolated)

- git clone --depth n <repo> = only clone history of depth n
  - repo's with extensive histories
- git clone --branch = clone specified branch (not the one HEAD points to)

1. runs git init <reponame>
2. copies all history from remote
3. creates '**origin**' ref which is remote connection back to source (in refs/remotes/origin)
4. initializes remote.origin.url and remote.origin.fetch config var

generally only need to run it once

What is origin?

- ‘origin’ is the alias for the url of original clone
  - Stored in .git/config file
  - Not created automatically with git init
  - Can add any alias to config manually or with
  - **git remote add <name> <url>**
    - git remote rm <name> = remove a connection

- Nothing automatically flows between repos even if you have a remote set up, it’s just an alias

Collaborating with git

Getting changes:
git pull or **git fetch**

Sending changes:
git push

Pushing branches to remote repos:

git remote add new-remote-repo <url>
git push new-remote-repo branch

delete a remote branch

git push origin --delete bad_branch **or**
git push origin :bad_branch

Pushing changes

git push <remote> <branch>

- `git push origin master` = pushing to master branch on origin (the server)
- Essentially the same as running `git merge master` from inside the remote repo
- change master to whatever branch you to want to push changes to
- A branch is not available to others unless you push it to your remote repo (git push origin <branch>)
- When you commit a file, you have committed to HEAD but it’s not in the remote repo yet. This is what push is for.
- Will not push if a fast-forward merge is not possible (--force)
- --all = push all branches

Example uses of git push

- Default use of push
  - Git checkout master; git fetch origin master = make sure local master is up to date
  - Git rebase –i origin/master =Rebase changes on top of central repo's copy
    - Also opportunity to clean up commits before sharing them
  - Git push origin master = send all commits on local master to central repo

- Amended force push
  - Git commit --amend
  - Git push --force origin master

- Deleting remote branch or tag
  - Git branch –D branch_name
  - Git push origin :branch_name

Steps you might need to take

- If you get request failed and url is https:
  - Use this url instead git@github.com:username/reponame
- If you get permission denied (publickey) you need to add the SSH key to your github account
  1. Create an SSH key:
     I. cd ~/.ssh
     II. ssh-keygen
     III. Hit enter for all the defaults, will create a new file called id_rsa.pub (or whatever you name it for non-default)
  2. Add key to your github account:
     I. Go to https://github.com/settings/keys
     II. New SSH key
     III. Copy and paste the contents of id_rsa.pub
  3. Type in consol:
     - ssh –T git@github.com

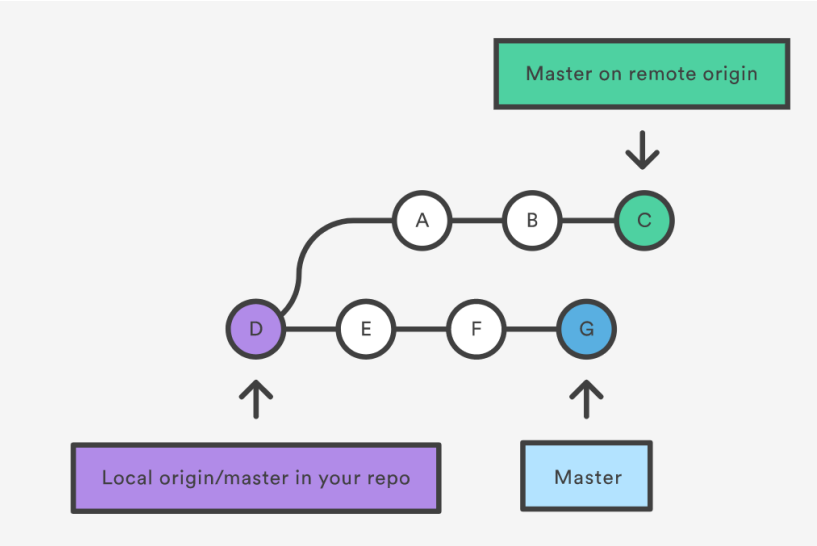
```
482 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git remote add origin git
@github.com:gsliwoski/giterdone
483 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[alias]
        ci = commit
[remote "origin"]
        url = git@github.com:gsliwoski/giterdone
        fetch = +refs/heads/*:refs/remotes/origin/*
484 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git remote -v
origin  git@github.com:gsliwoski/giterdone (fetch)
origin  git@github.com:gsliwoski/giterdone (push)
485 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

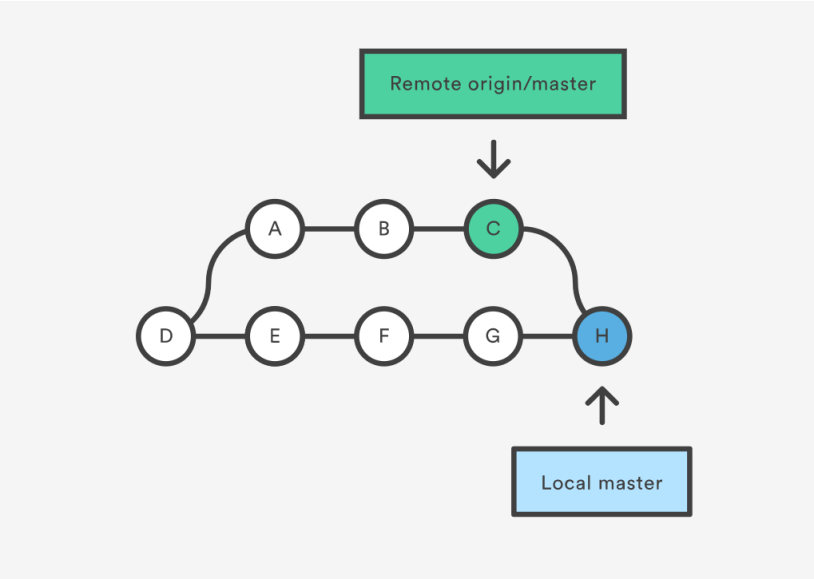
```
463 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git push origin master
Counting objects: 15, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (15/15), 1.64 KiB, done.
Total 15 (delta 3), reused 9 (delta 2)
remote: Resolving deltas: 100% (3/3), done.
To git@github.com:gsliwoski/giterdone.git
 * [new branch]      master -> master
464 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

Pulling changes

git pull

- often used without any parameters
- **--all** = pull all changes from server and merge into local repo with single command
- pull is a combination of git fetch followed by git merge.



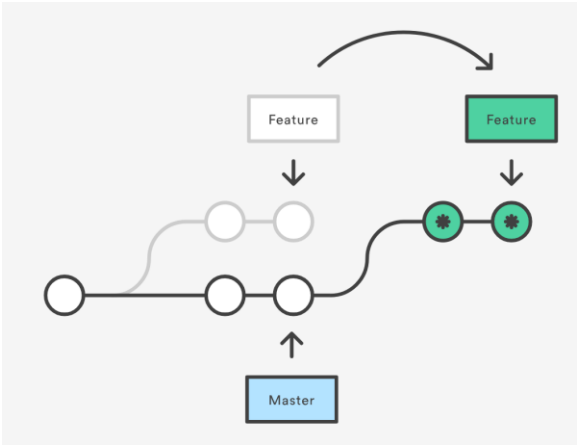


- Pull will download all changes from point where local and master diverged (E)
- New commit (H) is a new merge commit that contains contents of remote A-B-C commits and has a combined log message

```
487 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git pull
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> master

488 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git branch --set-upstream
-to=origin/master master
Branch master set up to track remote branch master from origin.
489 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git pull
Updating da710b6..8053b0f
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
490 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
AIclass.cplusplus         greghub.cplusplus  world_dominashun.cplusplus
flying_greghub.cplusplus  README.md
491 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat README.md
This is my own version of git. It's much better. And you can't crash the airplan
e either!
492 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

Keeping up-to-date with rebase

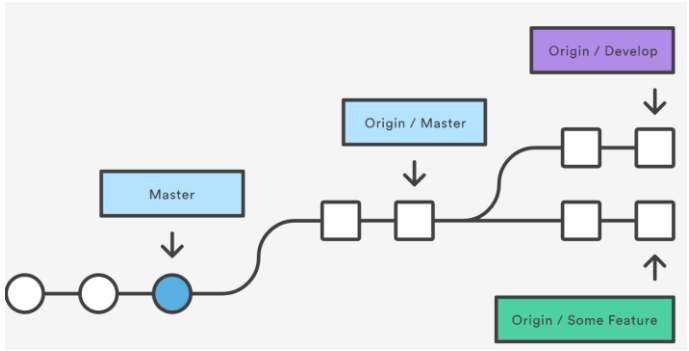
- Instead of pulling with merge, use rebase to keep things clean
  1. You clone and start a feature branch
  2. The origin master branch progresses while you're working
  3. You want to get updates but don't want to clutter history
     - Pull with merge would add unnecessary merge commits every time you pull
  4. Use git pull --rebase
     - To be careful, you can use –i for interactive rebase
     - Rebase often to avoid conflicts when origin master has gone too far



git config --global branch.autosetuprebase always = always rebase instead of merge with pull

git fetch (step 1 of git pull)

- Downloads changes but doesn't merge them
- Lets you see how central repo has progressed but doesn't force you to merge the changes into your repo.
- Fetched content must be explicitly checked out using git checkout
- Safe way to review commits before integrating them with local repo.
- All commits, local and remote are stored in the repo's .git/objects directory but remote and local branch commits are kept distinctly separate through use of branch refs. Local refs are stored in .git/refs/heads
- Remote branches are just like local branches except they map to commits from somebody else's repo and are prefixed by the remote they belong and are found in .git/refs/remotes
  - git branch –r (shows all branches: local are prefixed with origin, remote branches are prefixed with remote-repo/
- You can checkout a remote branch just like a local one but this puts you in a detached HEAD state (like checking out an old local commit).
- Ex: drop all local changes and commits:
  - git fetch origin
  - git reset --hard origin/master



Squares = commits downloaded from repo
Blue circle = your master

Forking

- Create your own public copy of a public repo
- Forking comes in because you typically have read access to another's public repo but not write access. Once you fork, clone, change, push, you can create pull request to write to it.
- Forking is another way of saving a clone or copy. Unlike branch, fork is independent from original repo. If original repo is deleted, fork remains. If you fork a repo, you get that repo and all of its branches.

Pull Request

- Ask another dev to merge one of your branches into their repo.
  - Essentially a comment thread attached to a feature branch.
  - Can be used when a dev gets stuck to ask for help from rest of team.
  - Formal code review, other devs can push changes
- Pull Request Workflow
  1. Dev creates feature branch in local repo
  2. Dev pushes branch to public repo
  3. Dev files a pull request
  4. Rest of team reviews code, discusses it, and alters it
  5. Project maintainer merges feature and closes pull request

Git LFS

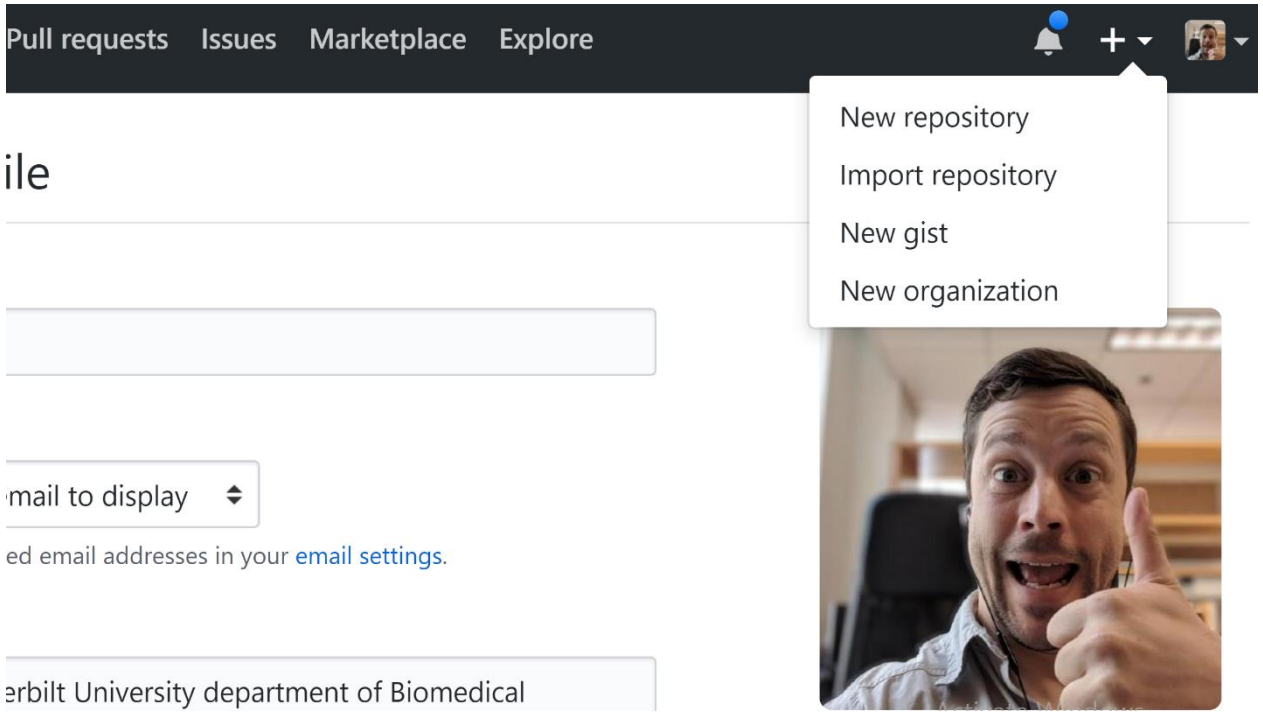
- Working with large files. Downloads relevant versions lazily (large files downloaded during checkout rather than during cloning or fetching)
- Replaces large files with pointer files.
- LFS is seamless so you use it without changing anything about your workflow.
- To use git LFS, use a git LFS aware host, most are, and have the git LFS command-line client installed:
- https://git-lfs.github.com/

Github

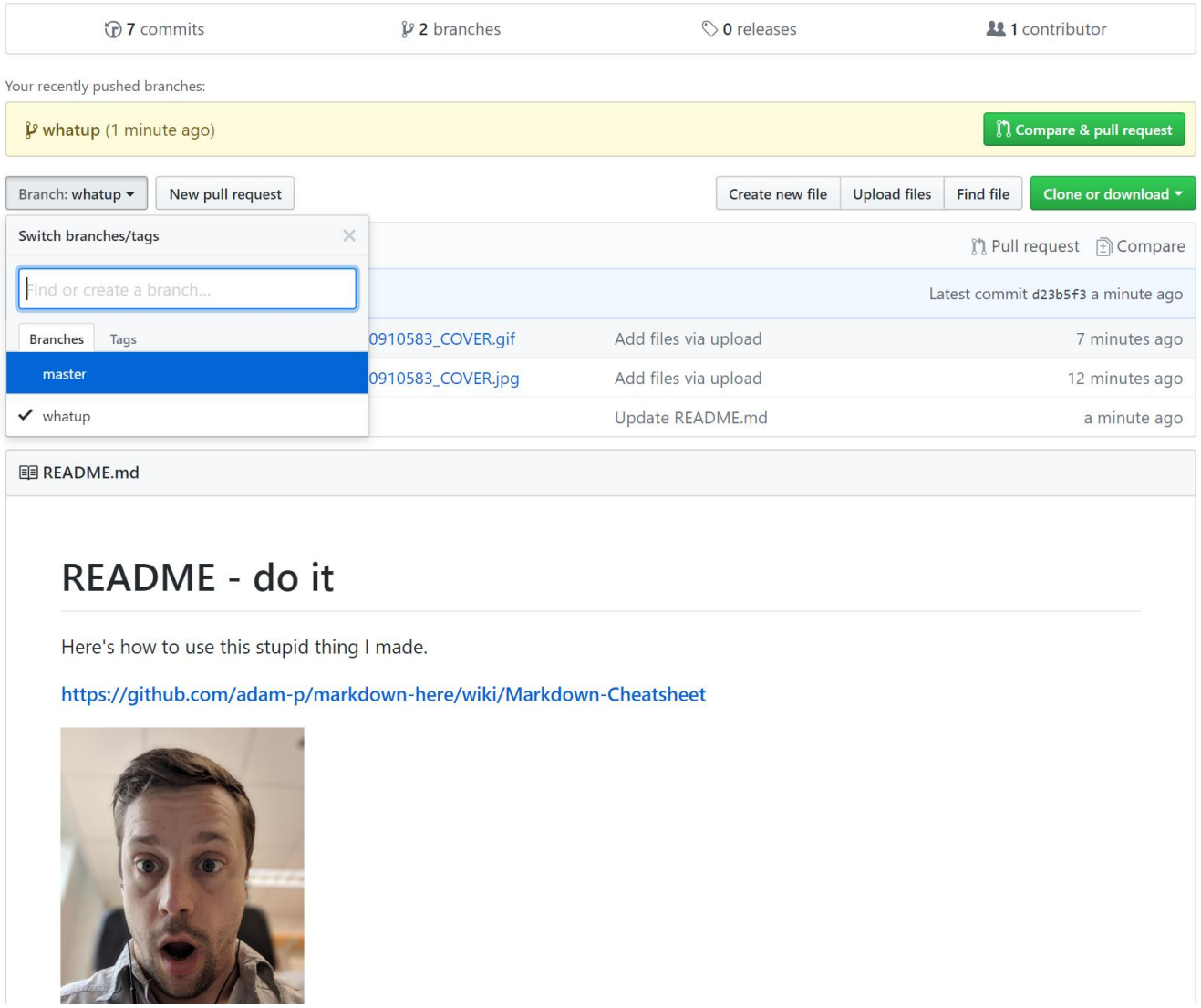
- Code hosting platform for version control and collaboration
- Repo = a single project, can contain folders and files, images, videos, datasets, anything your project needs
- Recommended to include a README or a file with info about your project (Github will prompt for a README when you create a new repo)

Creating a public repo on Github

- Click + next to avatar and select New Repository



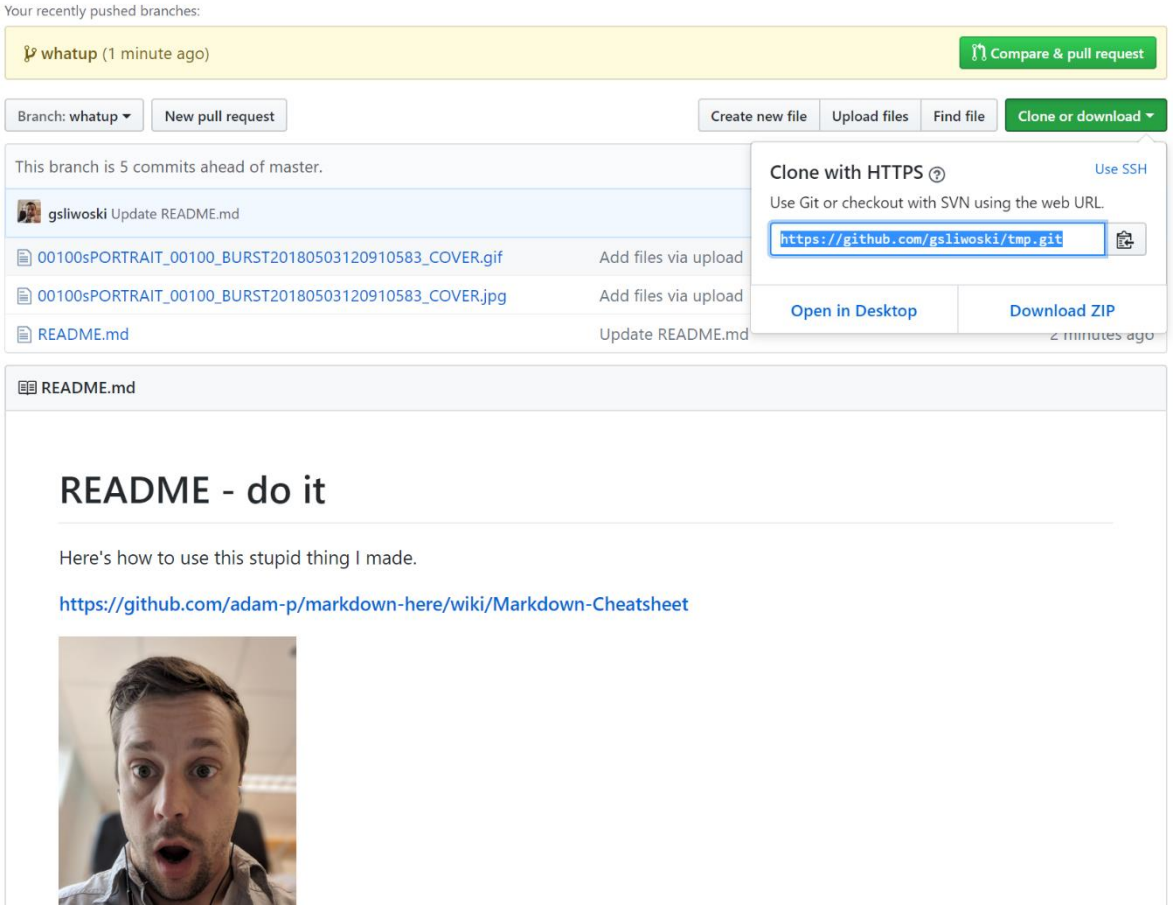
Doing things on Github



You can do most things from command line with buttons (create new branches, files, etc.

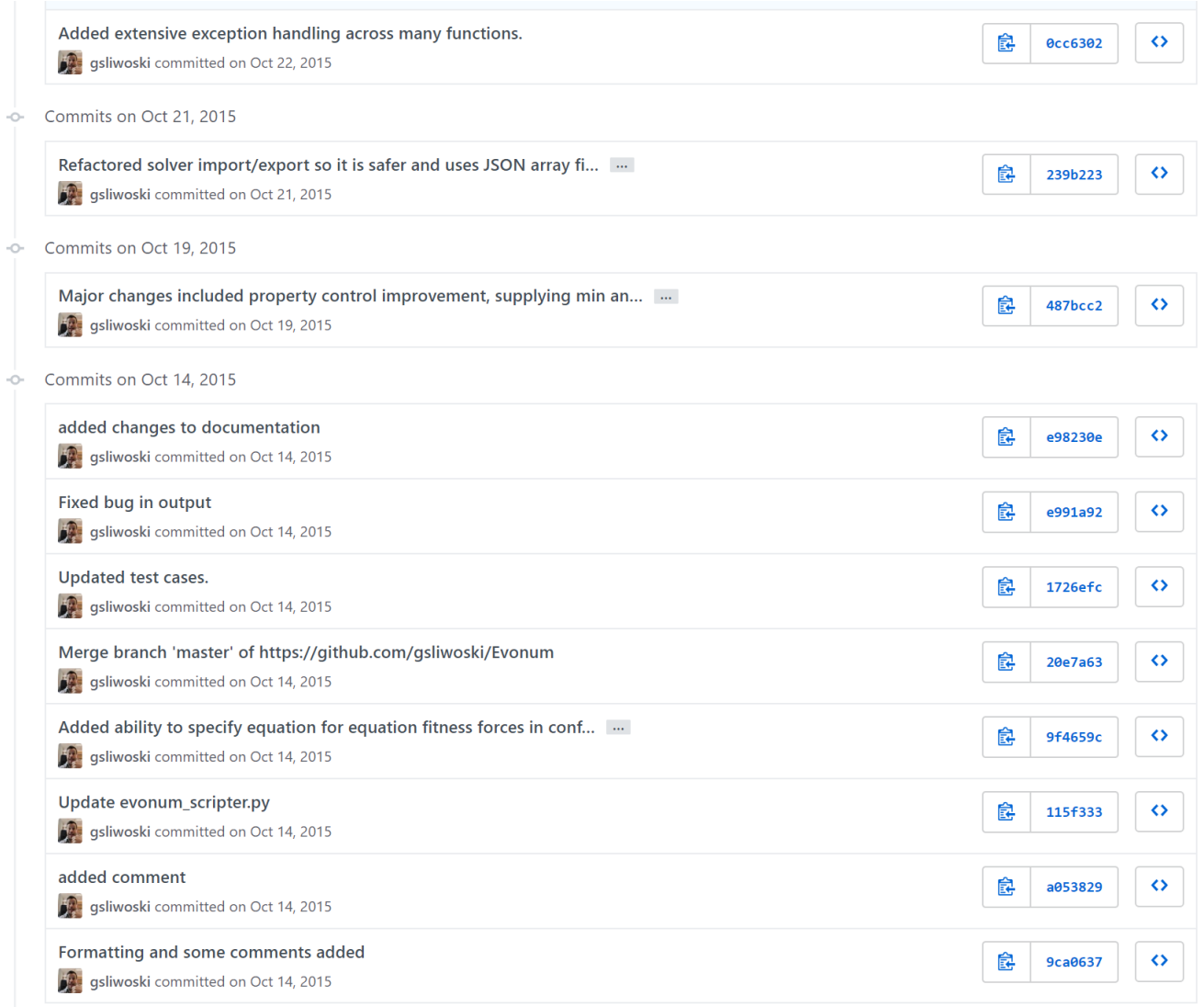
README is always displayed at project root
README uses markdown

Linking to a fresh github repo



This is your reference (origin when you clone)

Wow a fancy git log (AKA I prefer to click on stuff)



Wow you can look at files by clicking on them

Branch: master ▾ Evonum / evonum_terrarium.py Find file Copy path

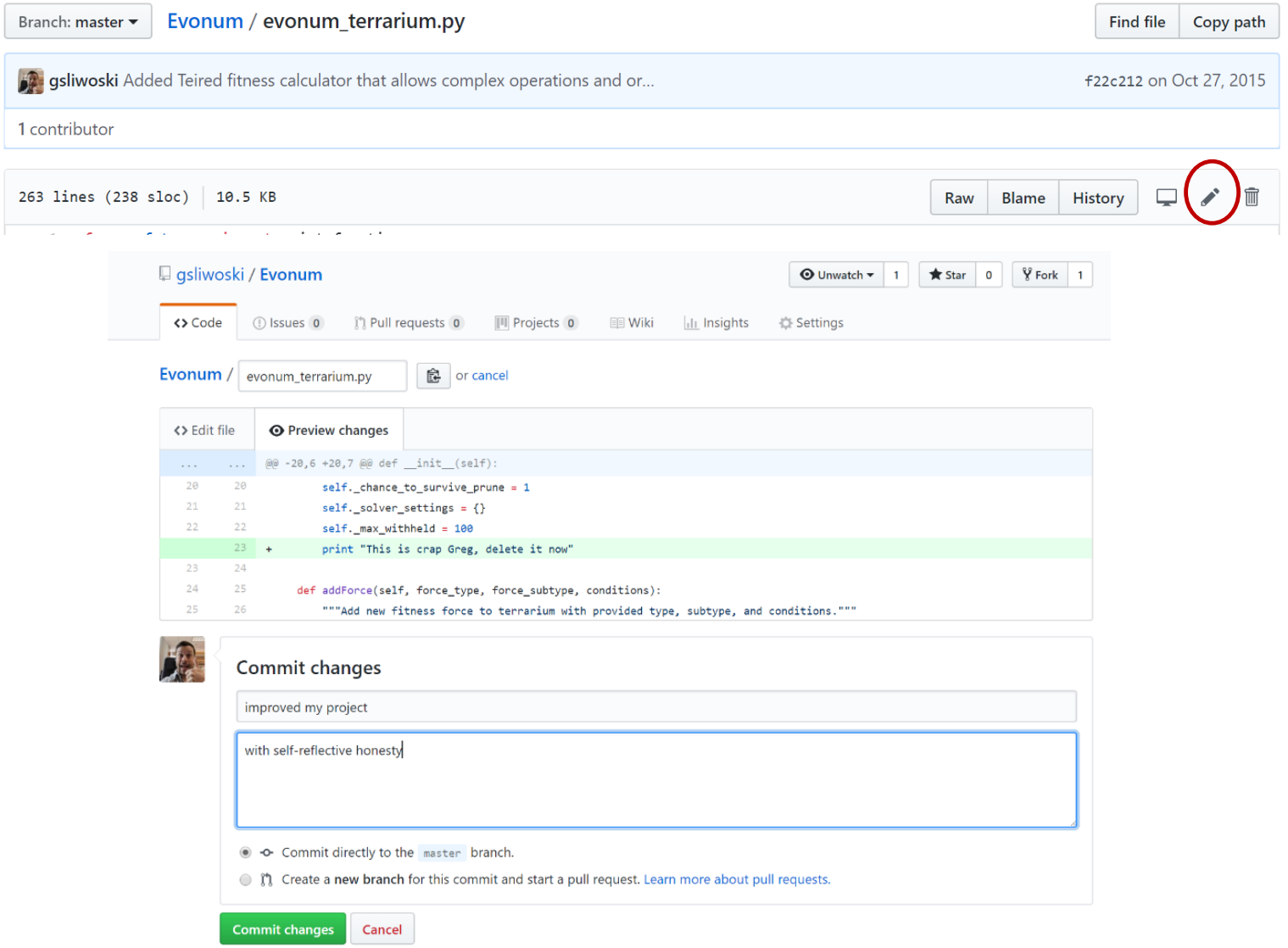
gsliwoski Added Teired fitness calculator that allows complex operations and or... f22c212 on Oct 27, 2015

1 contributor

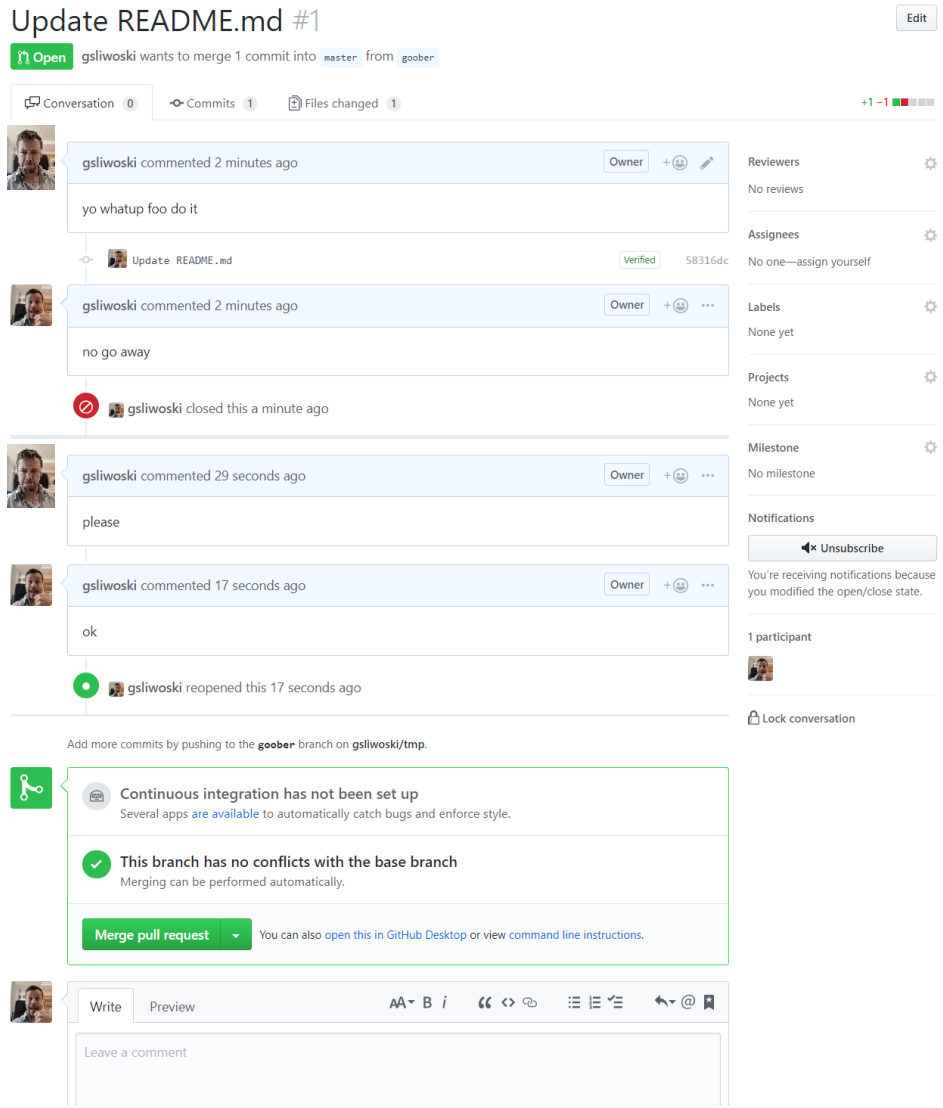
263 lines (238 sloc) | 10.5 KB Raw Blame History

```
from __future__ import print_function
import types
from evonum_fitness import *
from evonum_solvers import *
import json


def json_out(o):
    return o.exportDict()


class Terrarium(object):

    def __init__(self):
        self._current_day = 0
        self._forces = []
        self._solvers = []
        self._max_solvers = 100
        self._max_forces = 5
        self._chance_to_survive_prune = 1
        self._solver_settings = {}
        self._max_withheld = 100

    def addForce(self, force_type, force_subtype, conditions):
        """Add new fitness force to terrarium with provided type, subtype, and conditions."""
        if len(self._forces) >= self._max_forces:
            print ("Already at max forces. No force added.")
            return
        else:
            new_force = createFitnessForce(
                force_type, force_subtype, conditions)
        if new_force:
            self._forces.append(new_force)
        else:
            print("Error: failed to create force, no force added.")
```

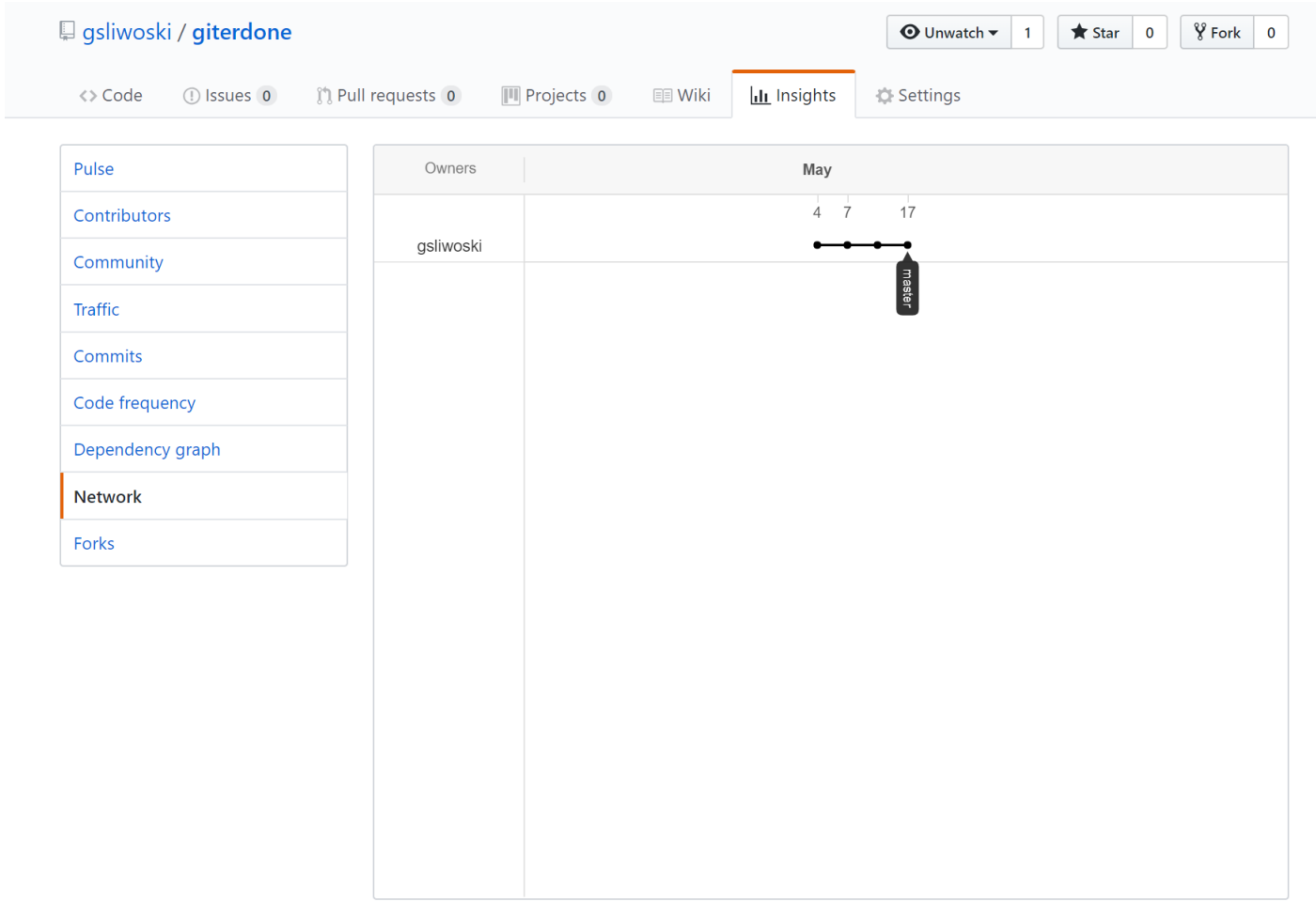
Wow you can edit files on github (aka vim sucks)



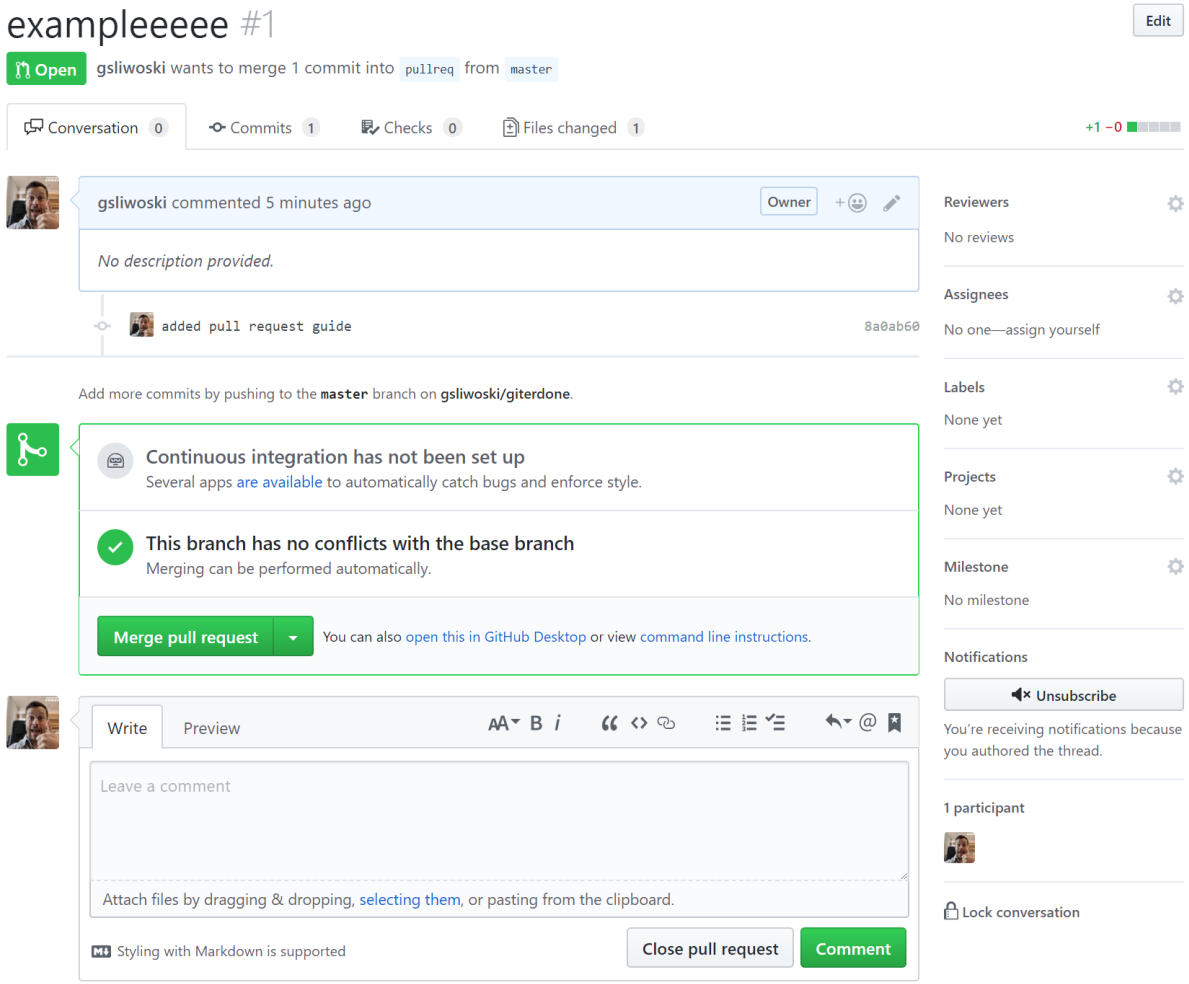
Pull requests are easy on github

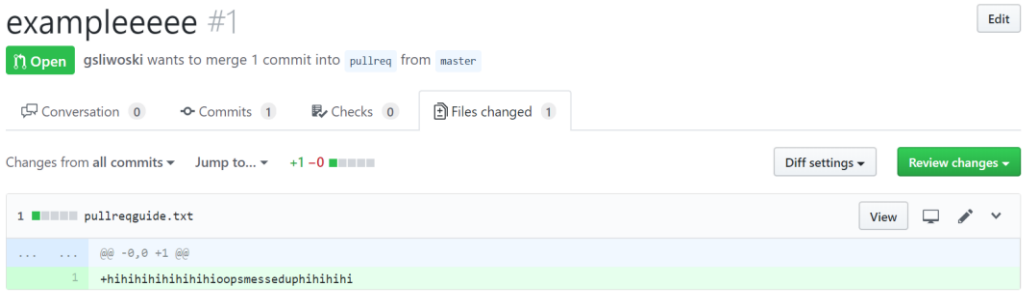


GITERDONE IS OFFICIAL



A pull request already?





Nice BUG!!!!! YOU FAILED

```
548 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git branch
* master
  sorry
549 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
AIclass.cplusplus         greghub.cplusplus  world_dominashun.cplusplus
flying_greghub.cplusplus  README.md
550 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git fetch origin pull/1/h
ead:pullreq
From github.com:gsliwoski/giterdone
 * [new ref]         refs/pull/1/head -> pullreq
551 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git branch
* master
  pullreq
  sorry
552 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git checkout pullreq
Switched to branch 'pullreq'
553 thallium:/dors/capra_lab/users/sliwosgr/giterdone% ls
AIclass.cplusplus         greghub.cplusplus  README.md
flying_greghub.cplusplus  pullreqguide.txt   world_dominashun.cplusplus
554 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat pullreqguide.txt
hihihihihihihihioopsmesseduphihihihi
556 thallium:/dors/capra_lab/users/sliwosgr/giterdone% sed -i 's:oopsmessedup::'
 pullreqguide.txt
557 thallium:/dors/capra_lab/users/sliwosgr/giterdone% cat pullreqguide.txt
hihihihihihihihihihihi
558 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git status
# On branch pullreq
# 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:   pullreqguide.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
559 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git add pullreqguide.txt

560 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git commit -m "fixed your
 nonsense"
[pullreq 85c0969] fixed your nonsense
 1 file changed, 1 insertion(+), 1 deletion(-)
561 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git push
fatal: The current branch pullreq has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin pullreq

562 thallium:/dors/capra_lab/users/sliwosgr/giterdone% git push origin pullreq
Counting objects: 5, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 284 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To git@github.com:gsliwoski/giterdone
   8053b0f..85c0969  pullreq -> pullreq
563 thallium:/dors/capra_lab/users/sliwosgr/giterdone%
```

gsliwoski / giterdone

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 1 | Projects 0 | Wiki | Insights | Settings

exampleeeee #1

Edit

Merged gsliwoski merged 1 commit into pullreq from master 14 seconds ago

Conversation 0 | Commits 1 | Checks 0 | Files changed 1 +1 −0

gsliwoski commented 14 minutes ago — Owner

No description provided.

added pull request guide 8a0ab60

gsliwoski commented 8 minutes ago — Owner

you HAVE A BUG IN YOUR CODE BUT I WILL FIX IT FOR YOU CAUSE YOU FAILED

gsliwoski merged commit 8a0ab60 into pullreq 14 seconds ago

Write | Preview

Leave a comment

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

Styling with Markdown is supported

Close pull request | Comment

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe
You're receiving notifications because you modified the open/close state.

1 participant

Github beyond repo's

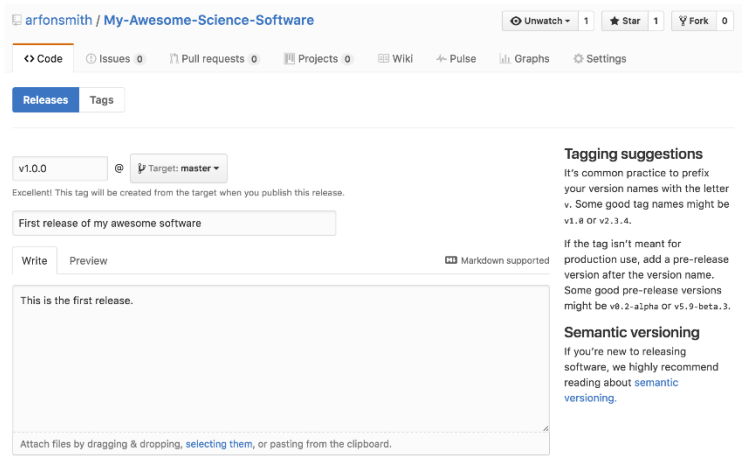
- Social Networking
  - **Follow** someone and get notified when they do stuff
    - Every profile has a follow button
  - **Watch** a specific project to get notified when it changes
    - Every project page has a watch button
  - **Explore** project that have been starred by people you follow, github staff, and general trending repos
    - The explore page is your news feed
  - **Inform** people on your project with the wiki that comes with every repo
- Bug tracking
  - Use **Issues** to keep track of tasks, enhancements and bugs
- **Citing your code**
  - Archive a github repo and get an assigned DOI citation reference

Citing your repo with **Zenodo**

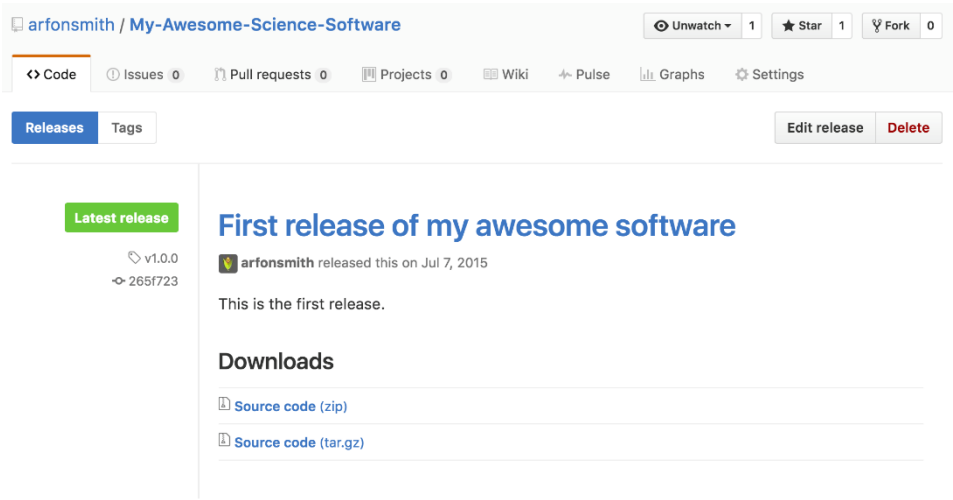
- Add a license to your repo to tell people how they can use it
  - www.choosealicense.com
- Log in to Zenodo with your github credentials
- Add the repo (must be public) so Zenodo can implement necessary webhooks for archiving and DOI-issuing
- Click the ‘on’ switch for your repo on Zenodo (this adds the webhook)
- Create a release (the release button on github)
  - By default, Zenodo takes an archive of your repo each time you create a new release
  - If you haven’t created a release yet, you will be asked to create one
  - There is a new release form you fill in (description, etc)

Creating a release and getting a DOI

1

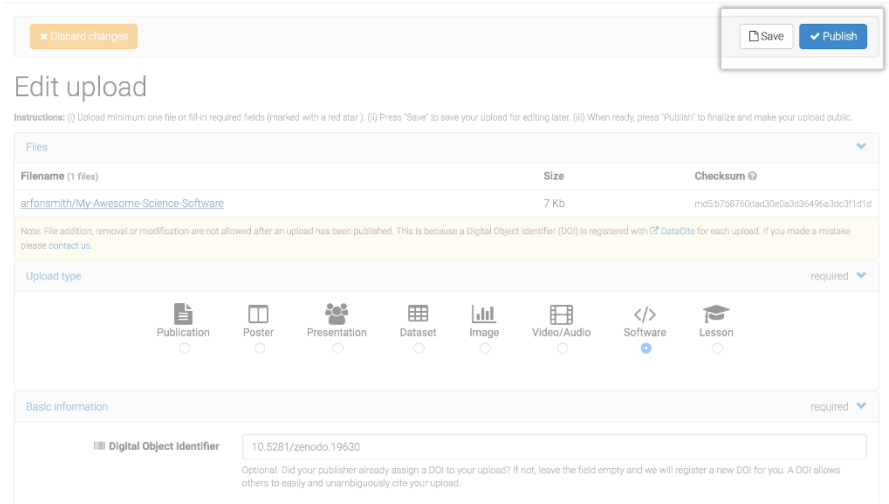


2



3. Zenodo was just triggered to archive the repo

4. Publish on Zenodo



5. Get your DOI