git Workflow Notes

Scott Kruger and Michel deMessieres

2020-01-12

This is not a tutorial.
Assumes basic merge and rebase knowledge.

Brief introduction to notation

- Svn uses *trunk* to denote main development branch, *tags* to denote released version, and *branches* to denote where individual development takes place
- Git has no standard rules regarding organization (*everything is a branch!)* so **convention** matters but is less standard (by default, you will have a "master" branch in new git repo)
- Most development teams have some notation corresponding to svn:
  - Branch to denote main development (fast-evolving stable version)
  - Branch to denote releases (slowly-evolving stable version)
  - All other branches (development branches where the real action is)

Comparison of conventions of two code teams

svn	PETSc	Trilinos
trunk	main	dev
stable	release	master
branches	devname/dev-goal	dev-goal
Example branch name:	`barry/fix-lapack-crash`	`fix-lapack-crash`

Notes on branch name conventions:

- For large projects, finding right branch can be hard (PETSc has ~500. Trilinos has more repos so fewer branches, but ~100 still common.) Some organizational structure seems useful and “devname/” is better than nothing.
- Use of “/” works well with gui’s (discussed later)
- devname/dev-goal seems to be useful even with multiple developers committing to a branch
- Hyphens are typically used over underscores for dev-goal

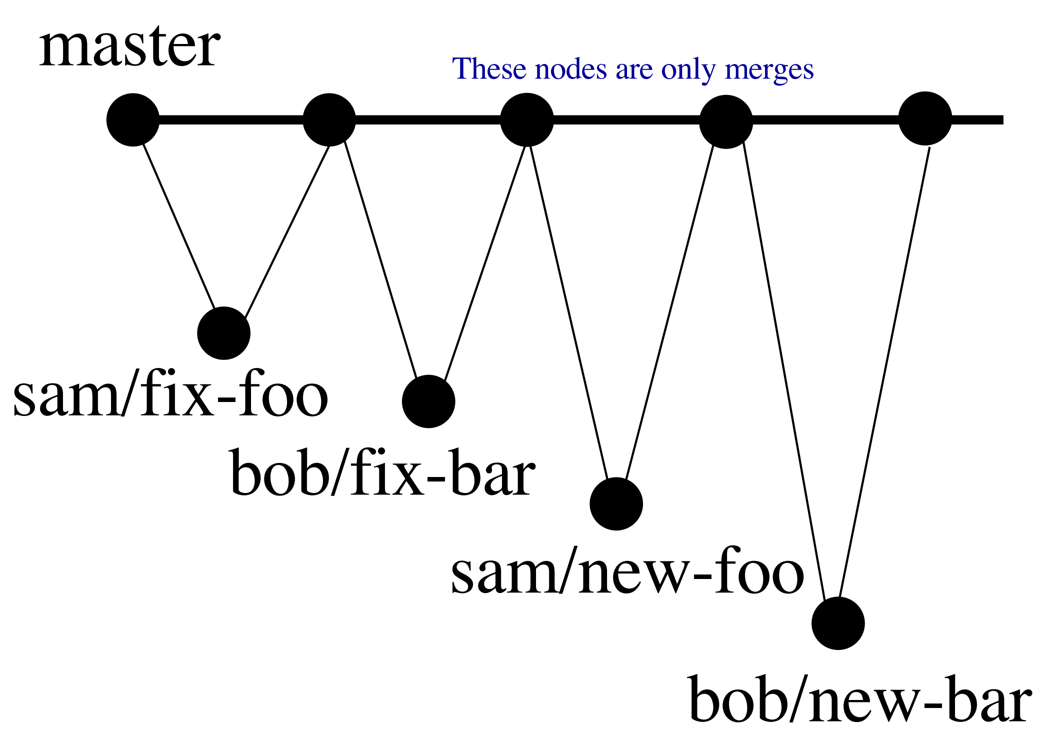
Ideal workflow has no commits other than merges on master/maint branch

- Every possible commit is first a branch, and then a merge back, even if you do not want to use merge request.
- Example typo fix:

```
git checkout master
git pull
git checkout -b scott/fix-typo-in-readme
<edit README>  # Could do this before the checkout -b
git commit -a -m'Fix typo'
git checkout master
git merge scott/fix-typo-in-readme
```

Ideal workflow has no commits other than merges on master/maint branch

- Simplest workflow in graphical view

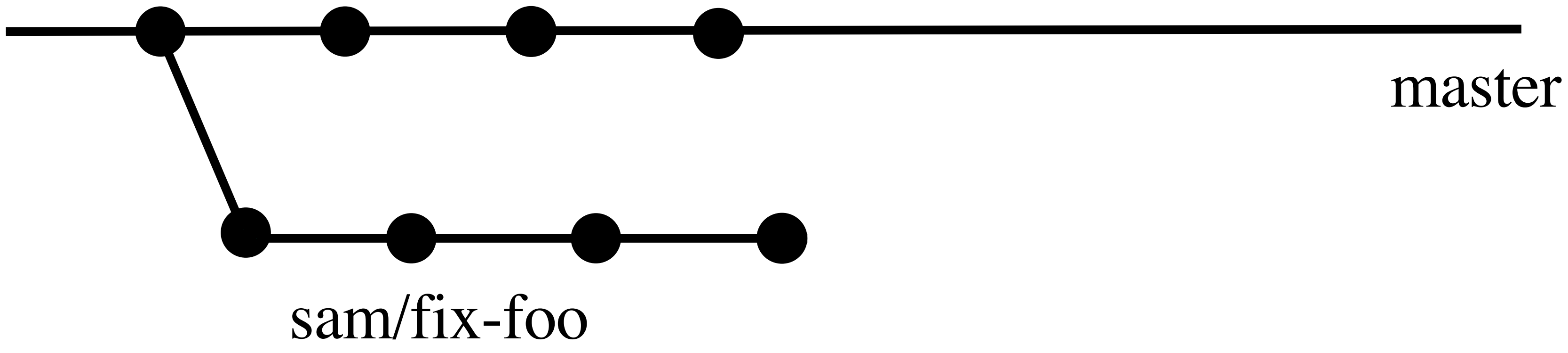


For this simple development, git log will show merge/dev/merge/dev/…

- Does not need to be single node in each branch
- Rule of thumb for # of nodes in dev branches: Make reasonable log message per file; i.e., `git log file` should always show relevant message

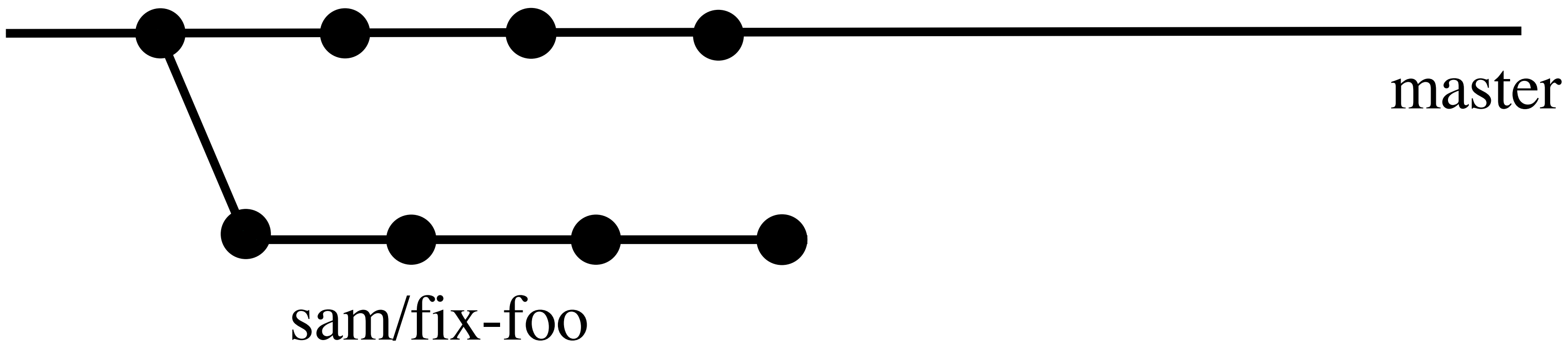
Making clean histories and clean trees

- Sam wants to open merge request to master.
- How should she best do this?



Making clean histories and clean trees

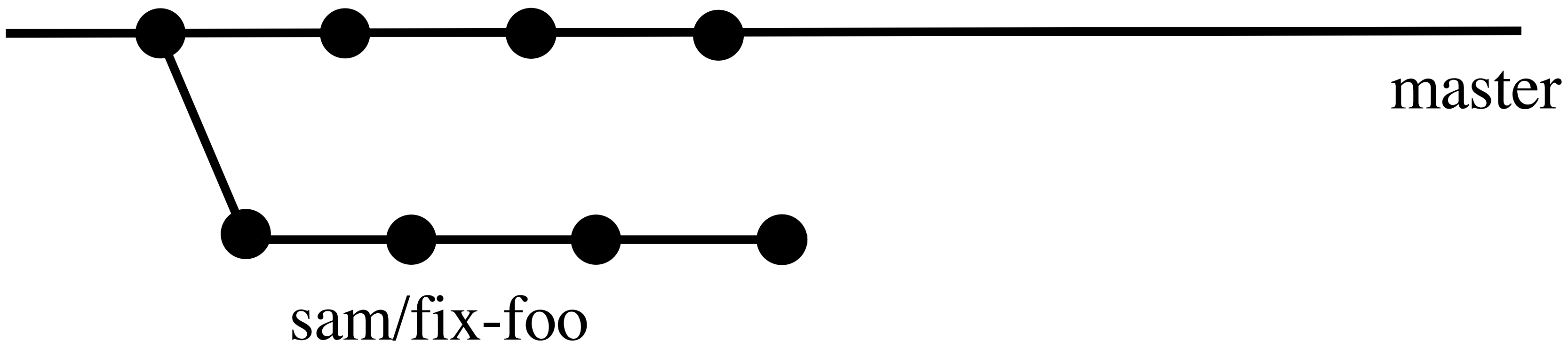
- Sam wants to open merge request to master.
- How should she best do this?



- First she should catch up to the last developments in master to make sure there are no conflicts.

Making clean histories and clean trees

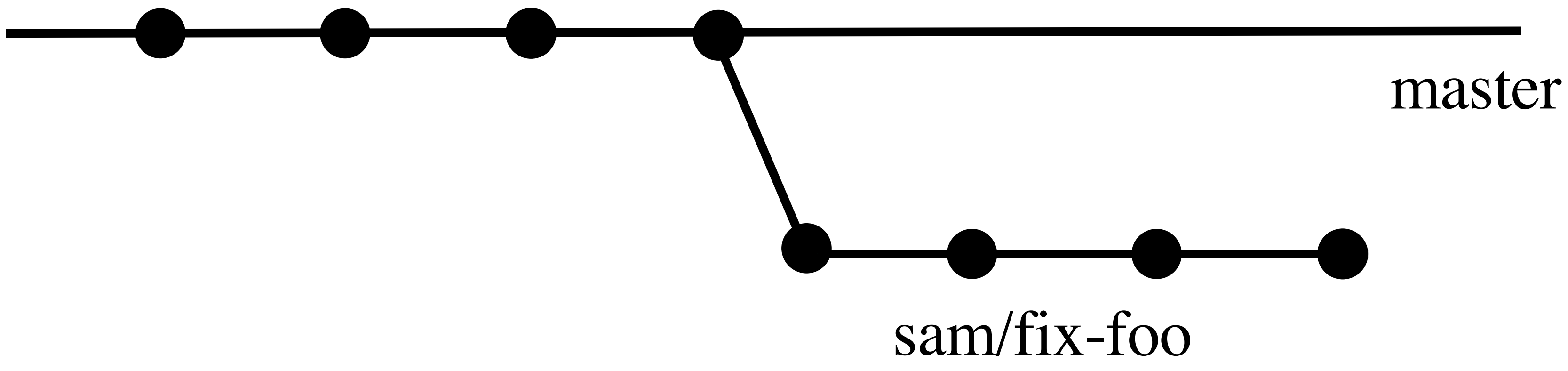
- Sam wants to open merge request to master.
- How should she best do this?



```
git checkout master
git pull  #make sure it's the latest
git checkout sam/fix-foo
git rebase master   # may require conflict resolution
```

Making clean histories and clean trees

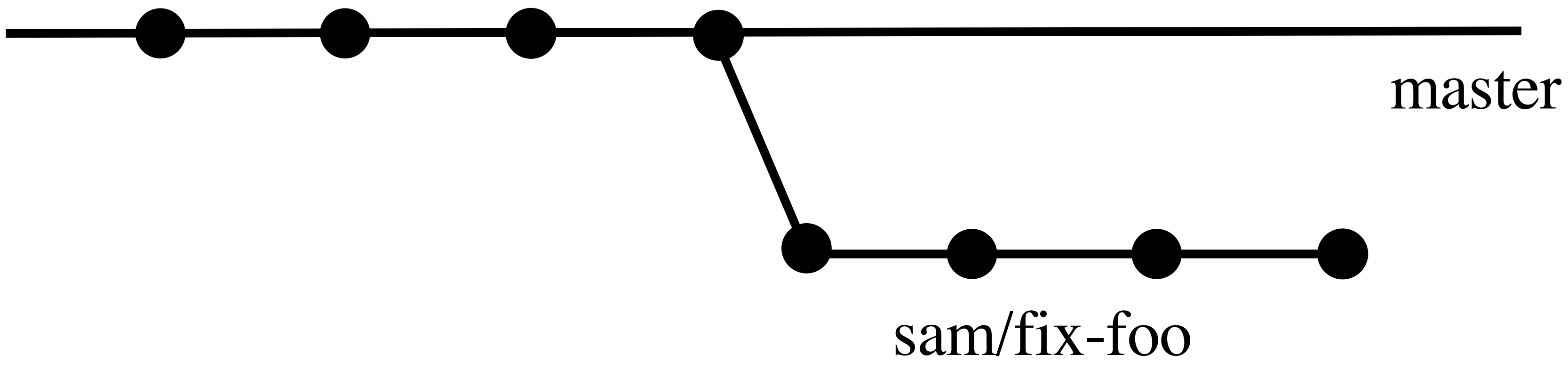
- Sam wants to open merge request to master.
- How should she best do this?



- After successful rebase
- But, two of those commits are just fixes of the prior. Need to squash

Making clean histories and clean trees

- Sam wants to open merge request to master.
- How should she best do this?

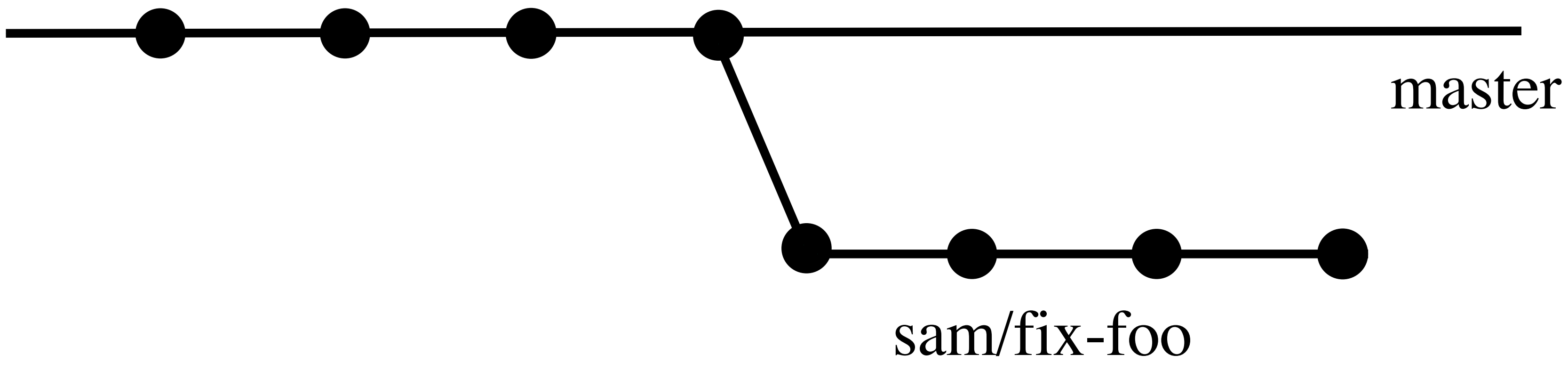


```
git rebase -i HEAD~4
```

-i => interactive
~4 => Last 4 from the latest commit (HEAD)

Making clean histories and clean trees

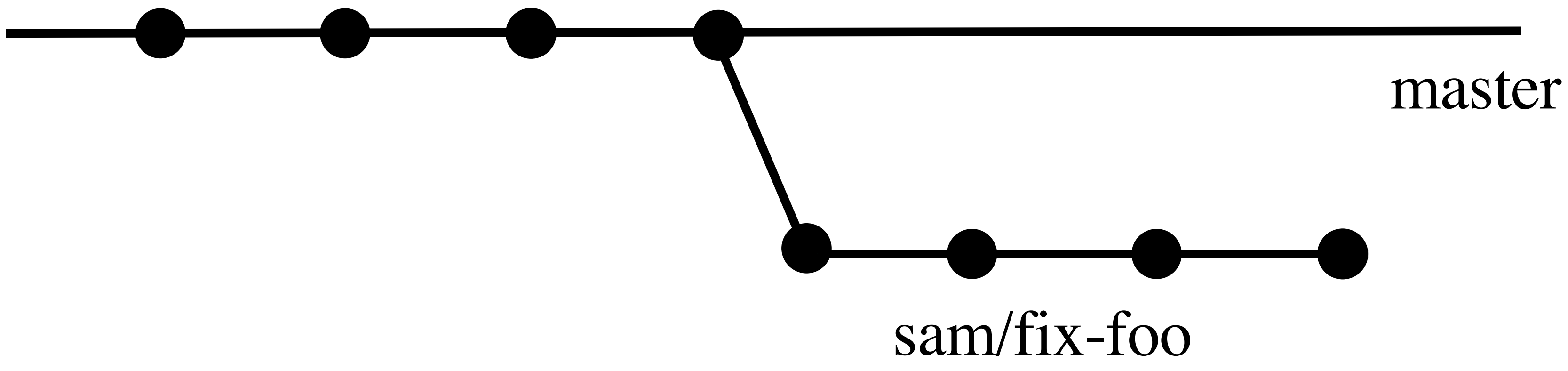
- Sam wants to open merge request to master.
- How should she best do this?



```
pick db986b2 Commit 1
pick 1fbcffd Commit 2
pick d7b34d0 Commit 3
pick 5d8c12a Commit 4
```

Making clean histories and clean trees

- Sam wants to open merge request to master.
- How should she best do this?

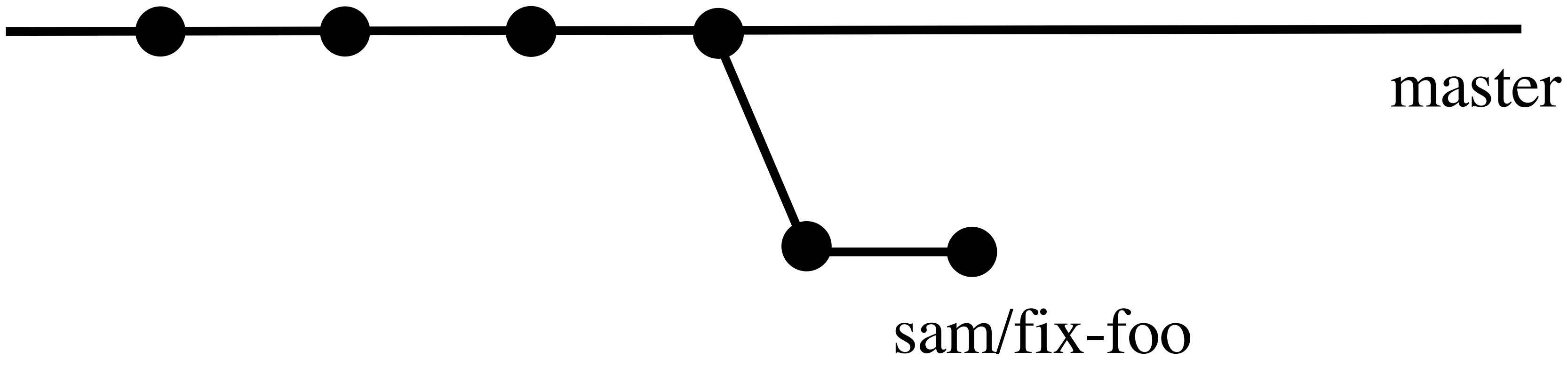


```
pick db986b2 Commit 1
squash 1fbcffd Commit 2
pick d7b34d0 Commit 3
squash 5d8c12a Commit 4
```

After editting the list, you will automatically be requested to edit log messages

Making clean histories and clean trees

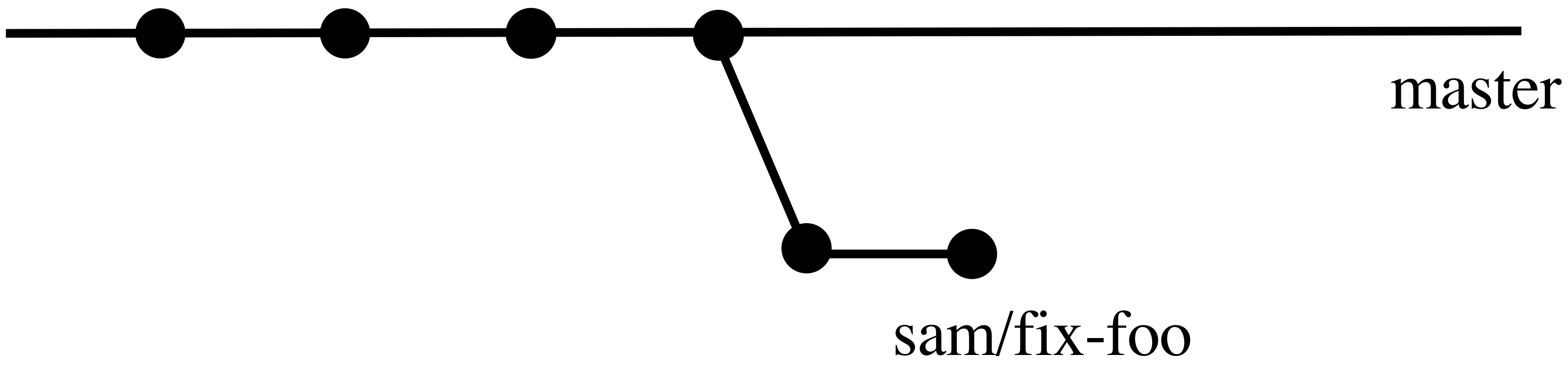
- Sam wants to open merge request to master.
- How should she best do this?



- Squash successful.
- Now need to open merge request

Making clean histories and clean trees

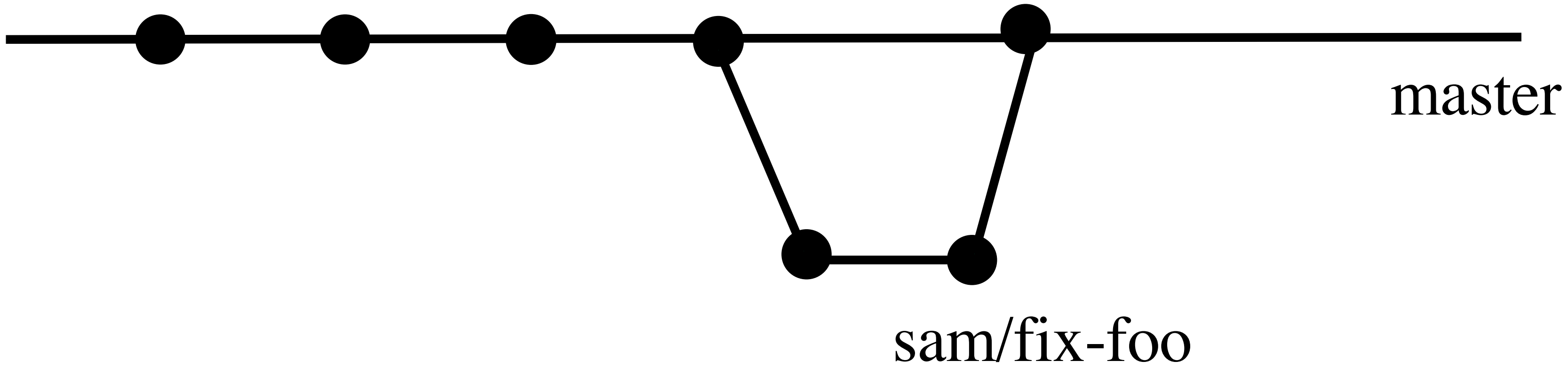
- Sam wants to open merge request to master.
- How should she best do this?



- Push to remote: `git push`
- Using ‘git checkout –b’ as above means requires to set origin name (can rename local branch from remote branch)
- It will show you command: just copy and paste
- It then will give URL for opening MR: just copy and paste

Making clean histories and clean trees

- Sam wants to open merge request to master.
- How should she best do this?

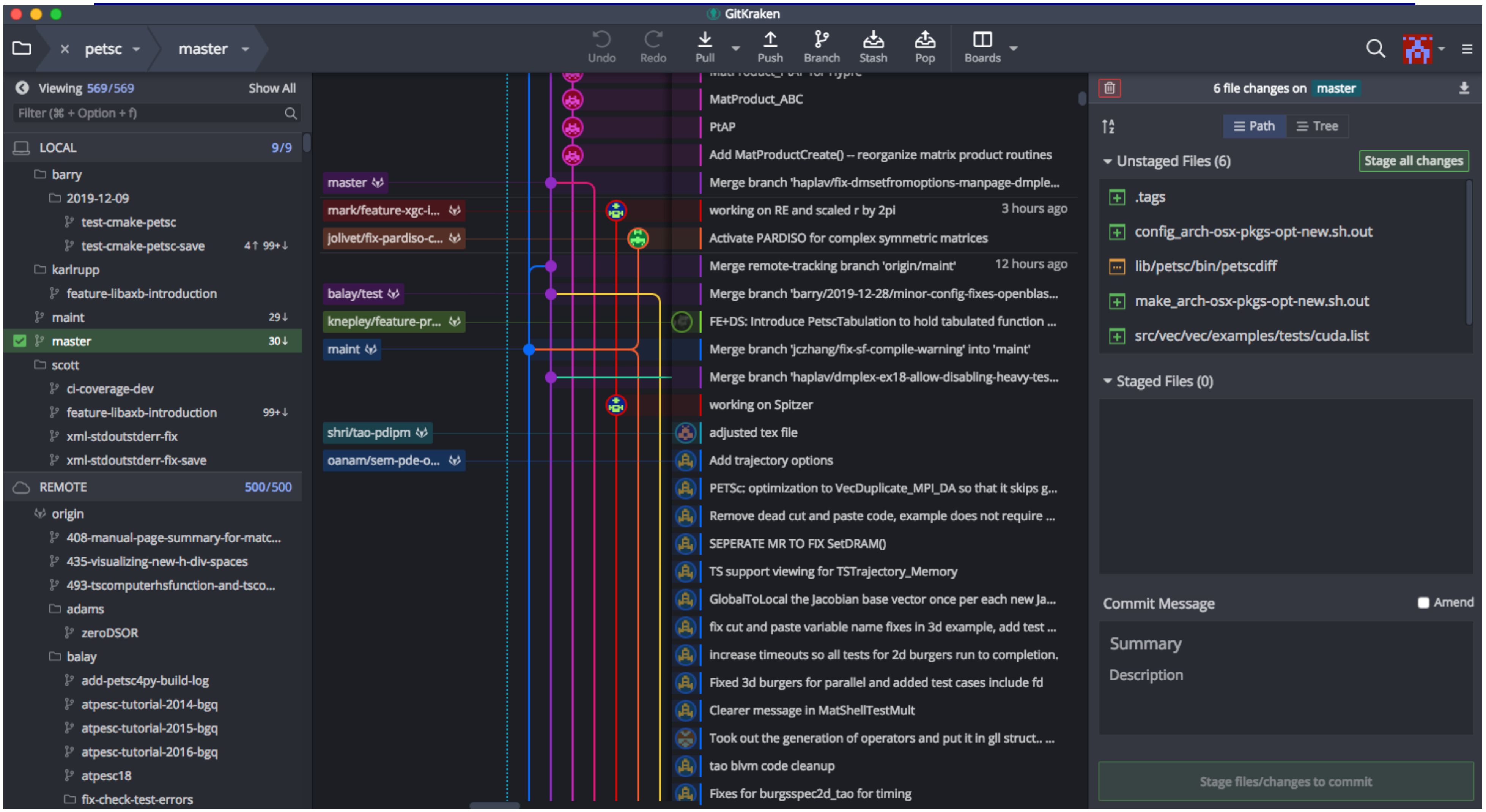


- After successful MR
- Resolving MR means doing testing (CI pipeline) and that can take time. May require new rebases.
- Don’t forget that if remote branch exists, the force pushes are required after squashing.

What to do when things go wrong

- GUI's can actually be useful
  - GitKraken: Commercial code that has free version
    - Good for people that like GUIs
  - gitk: Primitive GUI built on tk (often comes with git)
    - Excellent command line options make it faster and leaner
    - Still learning to use effectively
  - Both can show tree structure
  - When your branch shows a mess of the tree structure, then it can be hard to untangle
  - Bad branches usually result from merging instead of rebasing (rebase master in dev branches to catch up!)

GitKraken is pretty



Viewing 569/569 Show All

Filter (⌘ + Option + f)

LOCAL 9/9

- barry
  - 2019-12-09
    - test-cmake-petsc
    - test-cmake-petsc-save 4↑ 99+↓
- karlrupp
  - feature-libaxb-introduction
- maint 29↓
- ✓ master 30↓
- scott
  - ci-coverage-dev
  - feature-libaxb-introduction 99+↓
  - xml-stdoutstderr-fix
  - xml-stdoutstderr-fix-save

REMOTE 500/500

- origin
  - 408-manual-page-summary-for-matc...
  - 435-visualizing-new-h-div-spaces
  - 493-tscomputerhsfunction-and-tsco...
  - adams
    - zeroDSOR
  - balay
    - add-petsc4py-build-log
    - atpesc-tutorial-2014-bgq
    - atpesc-tutorial-2015-bgq
    - atpesc-tutorial-2016-bgq
    - atpesc18
  - fix-check-test-errors

Branch	Commit message	Time
	MatProduct_ABC	
	PtAP	
	Add MatProductCreate() -- reorganize matrix product routines	
master	Merge branch 'haplav/fix-dmsetfromoptions-manpage-dmple...	
mark/feature-xgc-i...	working on RE and scaled r by 2pi	3 hours ago
jolivet/fix-pardiso-c...	Activate PARDISO for complex symmetric matrices	
	Merge remote-tracking branch 'origin/maint'	12 hours ago
balay/test	Merge branch 'barry/2019-12-28/minor-config-fixes-openblas...	
knepley/feature-pr...	FE+DS: Introduce PetscTabulation to hold tabulated function ...	
maint	Merge branch 'jczhang/fix-sf-compile-warning' into 'maint'	
	Merge branch 'haplav/dmplex-ex18-allow-disabling-heavy-tes...	
	working on Spitzer	
shri/tao-pdipm	adjusted tex file	
oanam/sem-pde-o...	Add trajectory options	
	PETSc: optimization to VecDuplicate_MPI_DA so that it skips g...	
	Remove dead cut and paste code, example does not require ...	
	SEPERATE MR TO FIX SetDRAM()	
	TS support viewing for TSTrajectory_Memory	
	GlobalToLocal the Jacobian base vector once per each new Ja...	
	fix cut and paste variable name fixes in 3d example, add test ...	
	increase timeouts so all tests for 2d burgers run to completion.	
	Fixed 3d burgers for parallel and added test cases include fd	
	Clearer message in MatShellTestMult	
	Took out the generation of operators and put it in gll struct.. ...	
	tao blvm code cleanup	
	Fixes for burgsspec2d_tao for timing	

6 file changes on master

Path Tree

Unstaged Files (6) Stage all changes

- .tags
- config_arch-osx-pkgs-opt-new.sh.out
- lib/petsc/bin/petscdiff
- make_arch-osx-pkgs-opt-new.sh.out
- src/vec/vec/examples/tests/cuda.list

Staged Files (0)

Commit Message ☐ Amend

Summary

Description

Stage files/changes to commit

Good way of recovering

- Local branches need not match remote branch name
- Multiple local branches can point to same remote
- Rename your local branch (m = move):
  - `git branch -m sam/fix-foo-bad`
- Cherry-pick the good stuff from bad:

```
git log
<copy hash of commit you want>
git checkout master
git pull  #make sure it’s the latest
git checkout -b sam/fix-foo
git cherry-pick <paste hash>
git log  # Check and make sure it is clean.  Or use gui
git push -f # Force the update
```

Working with other developers

- If other people are pushing to your branch, then beware of \`git pull\`.
- Good method: Always pull while in master, and look and see if your branch has new commits to avoid surprises.
  - Can also just do fetch (pull = fetch + merge) but I tend to prefer pull in main since I want main to be up-to-date anyway
- If your pull showed a forced push on your branch, then saving your own local branch and checking out fresh branch may be a good idea
- If while doing a merge (or rebase) it seems bad, \`git merge –abort\`, then a branch move and cherry-pick might be easier.

Final thoughts with random coherency

- Having multiple local branches being able to point to a single remote branch is very useful and not obvious in how it impacts workflow. It allows you to save stuff permanently (more useful than stash IMO)
  - Specifically: ‘git branch –m’ and ‘git cherry-pick’ can really save a lot of time when things are fubar’d.
- Having multiple remotes is also useful, and critical for collaborating with teams where you are not a member; e.g., trilinos only accepts pull requests from forks so I need a remote from the main repo and a remote from our fork to work effectively
- Looking at .git/config can help with the above to understand where your local branches are pointing to.
- Picking a complicated repo and looking at tree structure using GUI is educational.