

Lecture 10

Stash, Blame, Undoing, and Visual Git Tools

Schedule

- Only a few classes left 😞
- Today (4/12): Stash, Blame, Undoing Things, SourceTree
- 4/19: Carnival! – no class
- Last Lecture (4/26): Reflog, Plumbing and Porcelain
- Last Class (5/3): Final Exam

Scenario: you want to switch branches, but you have uncommitted changes

```
:( Andrew@Gengar ~/temp
04:08 PM (coolfeature)$ git checkout master
error: Your local changes to the following files would be overwritten by checkout:
A
Please, commit your changes or stash them before you can switch branches.
Aborting
```

What if you don't want to commit?

git stash

Example use:

git stash



- Makes a “pseudo-commit” and puts it on a **stack** of stashed pseudo-commit.
- Message for stash is “WIP on <*branchname*>...”
- Use git stash save <message> to store stashes with better messages

git stash list

Example use:

git stash list

- Lists all stashed changes on the stash stack



git stash apply (stash@{<depth>})

Example use:

```
git stash apply stash@{2}
```

- Reapplies the stashed change at the specified depth, if given.
- Depth is just another way of choosing from a list of saved stashes



git stash pop

Example use:

git stash pop

- Reapplies the top stashed change and removes it from the stash stack.

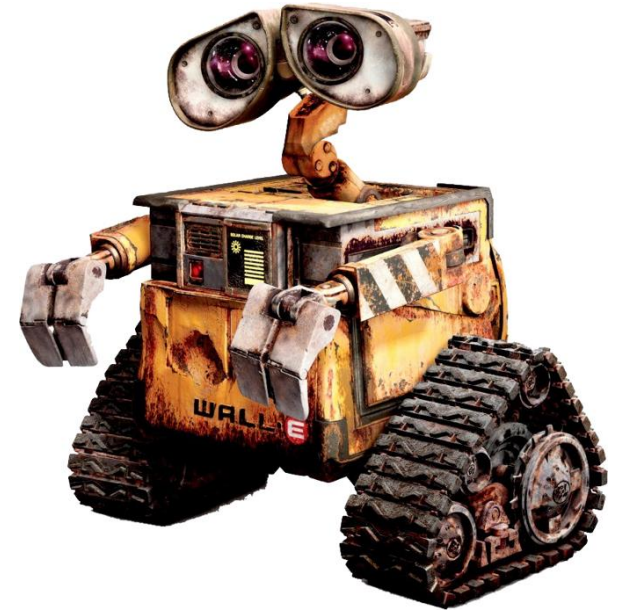


git stash drop (stash@{<depth>})

Example use:

```
git stash drop stash@{2}
```

- Removes the stashed change at the specified depth, if given.



git stash show (-p) (stash@{<depth>})

Example use:


```
git stash show stash@{2}
```



- Show details about the stashed change at the specified depth, if given.






File History on Github

Branch: master ▾ Autolab / app / models / annotation.rb Find file Copy path

 icanberk Annotations for PDF submission ce723aa on Jul 8, 2015

2 contributors  


Executable File | 26 lines (23 sloc) | 564 Bytes Raw Blame **History**   







```
1 ##
2 # Annotations are Submission and Problem specific.
- - - - -
```

- Same as `git log <filename>`

git blame <filename>

- Shows which commit last modified a specific line

Autolab / app / models / annotation.rb 

Commit	Author	Time	Line	Code
100755			26 lines (23 sloc)	564 Bytes
	Rubocop'd A models	2 years ago	1	##
			2	# Annotations are Submission and Problem specific.
			3	# Currently, they are just text, but it would be nice if they could be used in
			4	# score calculations in the future.
			5	#
	Initial commit for open source Autolab	2 years ago	6	class Annotation < ActiveRecord::Base
			7	belongs_to :submission
			8	belongs_to :problem
			9	
	Annotations for PDF submission	2 years ago	10	validates :comment, :filename, :submission_id, presence: true
	Initial commit for open source Autolab	2 years ago	11	
	Fixes for viewFeedback	2 years ago	12	def as_text
	Ran rubocop --autocorrect	2 years ago	13	if value
			14	if problem

Undoing Things: Commit

Just the message: `$ git commit --amend`

Files too:

`$ git reset --soft HEAD~` (moves changes back to index)

<make fixes>

`$ git commit ...`

Undoing Things: Merge

The feature branch hasn't moved, just need to move master back

```
$ git reset --hard HEAD^
```

Why "^" and not "~"?

^ = first parent

^2 = second parent

^n = nth parent (waaaay to complicated, google Octopus merge)

Undoing Things: Rebase

Be Safe: Drop a backup branch before you rebase

<On branch feature>

```
$ git branch feature-before-rebase
```

```
$ git rebase master
```

<crap I don't want this>

```
$ git reset --hard feature-before-rebase
```

What we've done so far

- Used git in the terminal
 - Best for learning (in my opinion!)
 - Hopefully no magic!
- Used Github's UI
 - Great for collaborating with others
 - Also check out [GitLab](#) and [Bitbucket](#)
 - Can't do everything



Why not just use the command line?

- Using vim for commit messages and interactive rebase
- <<<<<<<<<<<<<
=====
- >>>>>>>>>>>>>
- git log --graph is impossible to read
- Interactive add (git add -i) UI is horrible
- Having to remember and type commands (aliases help with this)



Atlassian SourceTree

GUI for Git

<https://www.sourcetreeapp.com/>

Produced by Atlassian

You need an Atlassian account to install SourceTree for some reason



SourceTree's powers

- Almost everything in the terminal can be done just as easily as in SourceTree
- The UI is often more intuitive
- SourceTree shows you things like graphs and diffs along the way without you having to ask

