How To Use Git

Advanced:
Tags & Branches

Mary Kate Trost
July 8, 2011

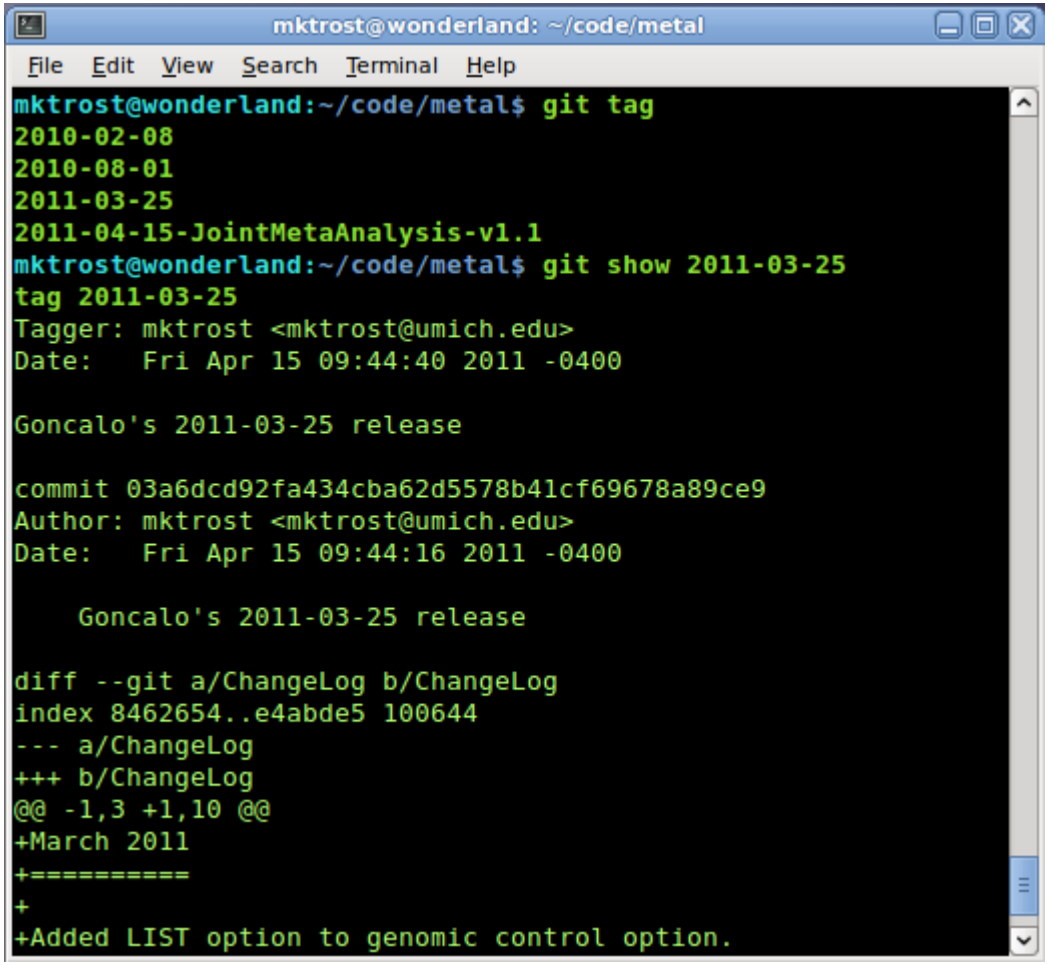
Create a Version (Tags)

- When releasing, want to mark that version so you can go back to it
- Create a tag
  - git tag -a *tagName* -m '*tag description*'
  - Example:

    git tag -a v1.1 -m 'version 1.1 contains the first version'

Git Tags

- List tags
  - git tag
  - git tag -l *v1.1.**
- Get info on a tag
  - git show *tagName*
- Push to remote
  - Not automatically pushed to the *remote*
  - git push origin *tagName*



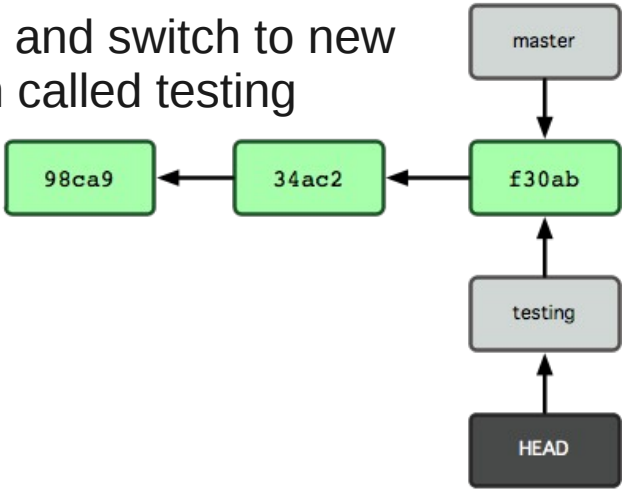
Branching

- Start from a certain version and make modifications separate from the main version
  - Allows you to develop changes and store intermediate files prior to putting them in the version everyone else sees (a form of backup)
  - Allows you to collaborate with a couple people without affecting everyone else
- See what branches there are
  - git branch
  - Current branch marked with '*'

```
$ git branch
  master
* pileup
```

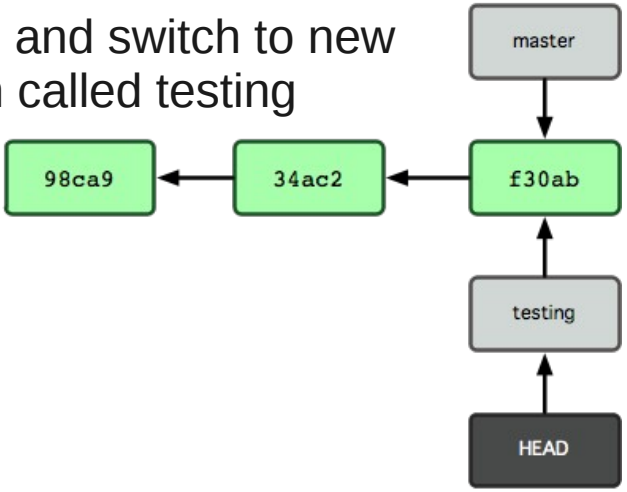
Branch Concept

Create and switch to new branch called testing

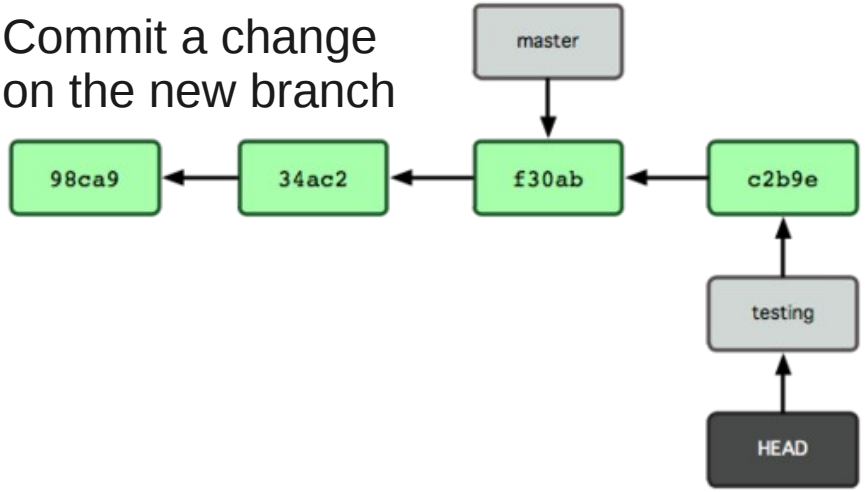


Branch Concept

Create and switch to new branch called testing

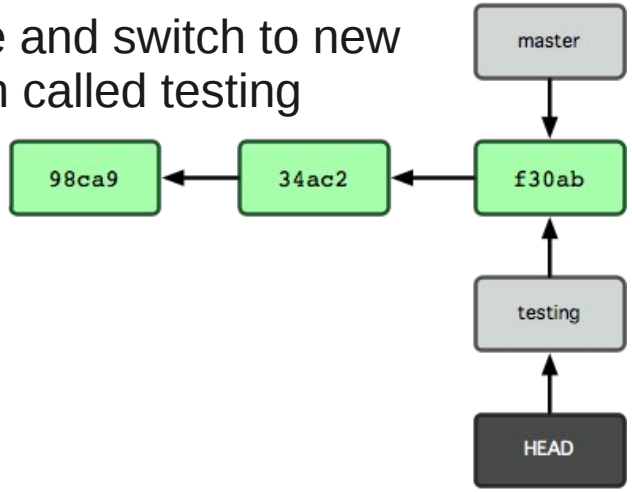


Commit a change on the new branch

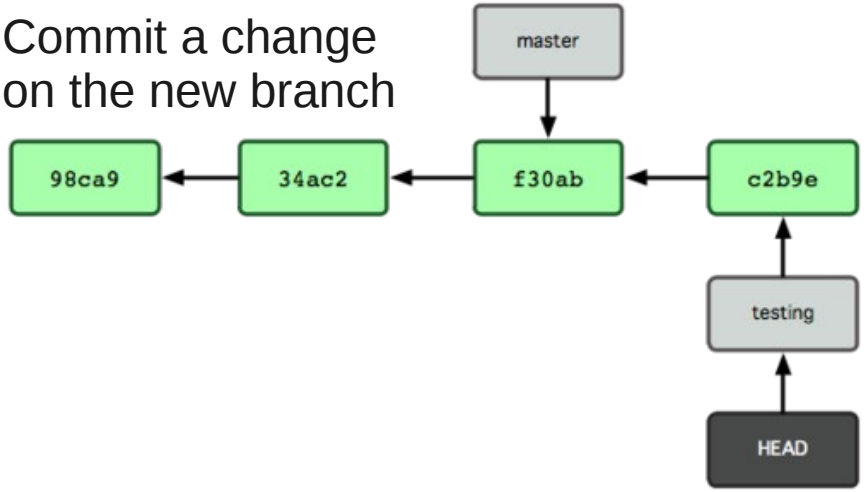


Branch Concept

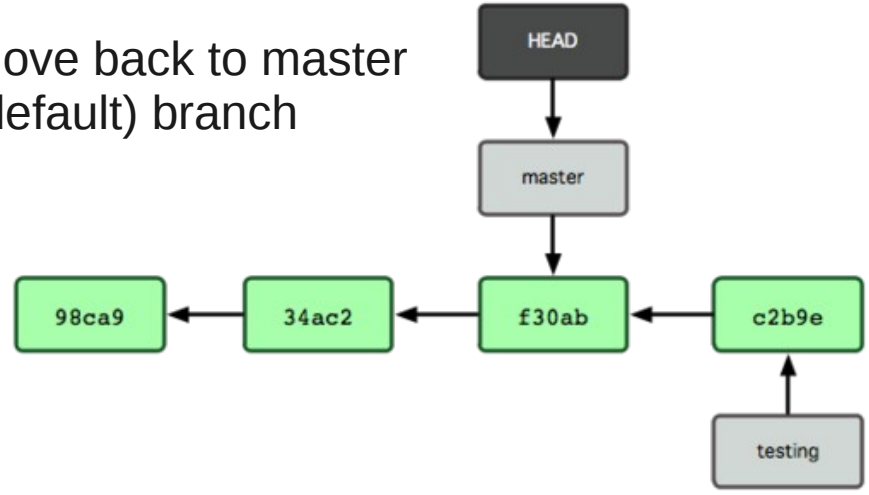
Create and switch to new branch called testing



Commit a change on the new branch

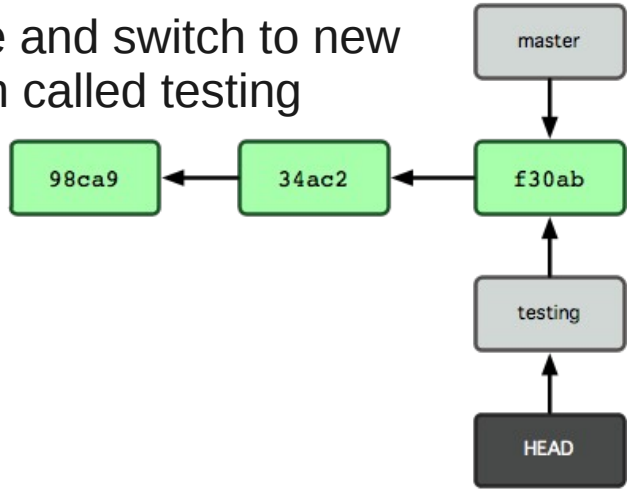


Move back to master (default) branch

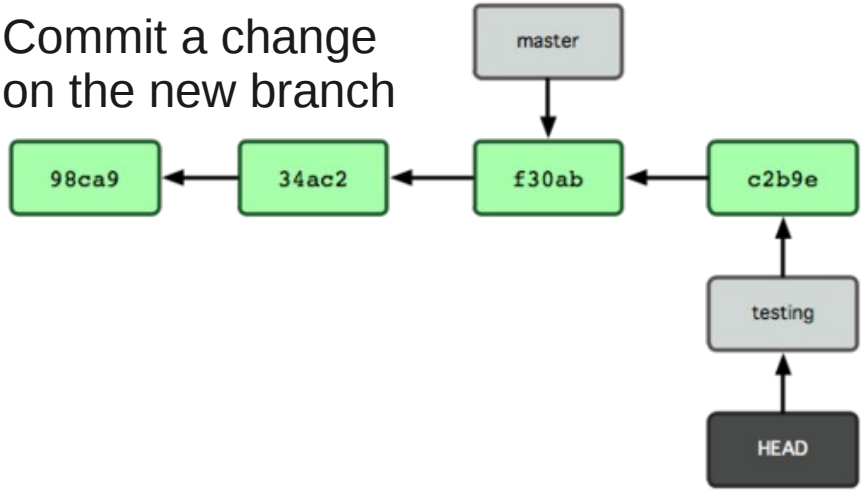


Branch Concept

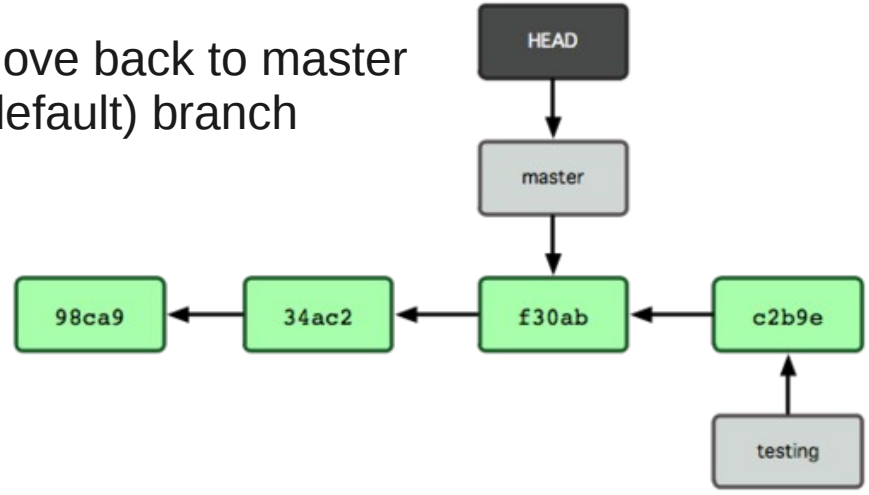
Create and switch to new branch called testing



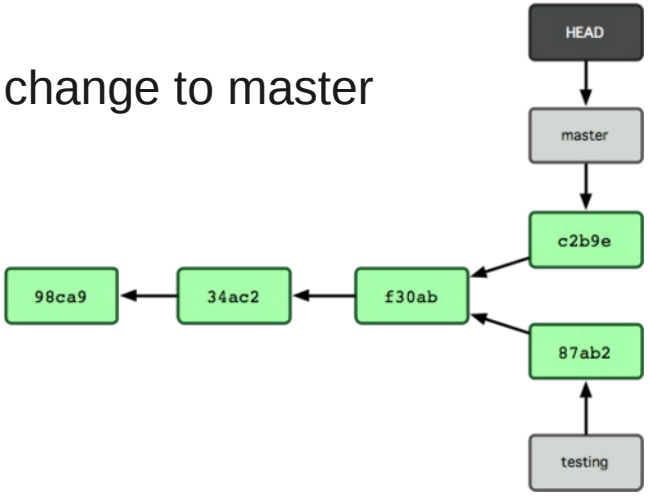
Commit a change on the new branch



Move back to master (default) branch



Commit a change to master

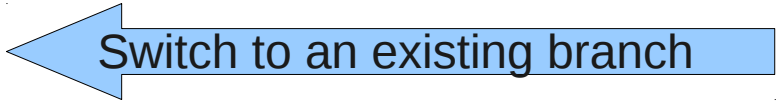


From: http://progit.org/book/ch3-1.html

Switching Branches

- Uncommitted changes will end up in the new branch
  - Commit everything that goes in the current branch before switching
  - Branch checkout fails if uncommitted changes conflict with the branch
- git checkout -b *branchName*

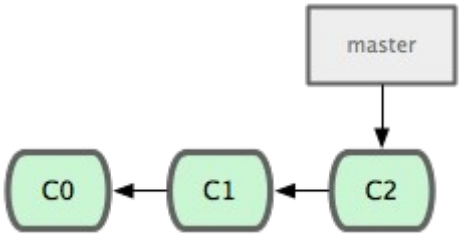
  - Creates starting from the currently checked out branch
    - May need to switch branches before creating a new one
- git checkout *branchName*



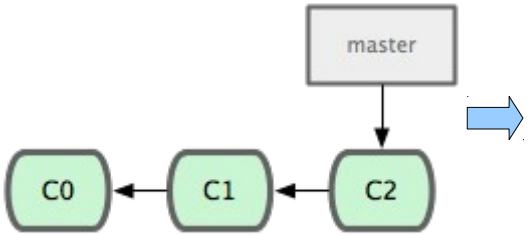
Branch Merging

- Checkout the branch you want to merge into
  - git checkout *branchMergeInto*
  - Often: git checkout master
- Merge the other branch into it
  - git merge *branchMergingFrom*
  - ”Fast forward” - no divergent work, just moves the pointer to the latest commit on the other branch

Branch Example

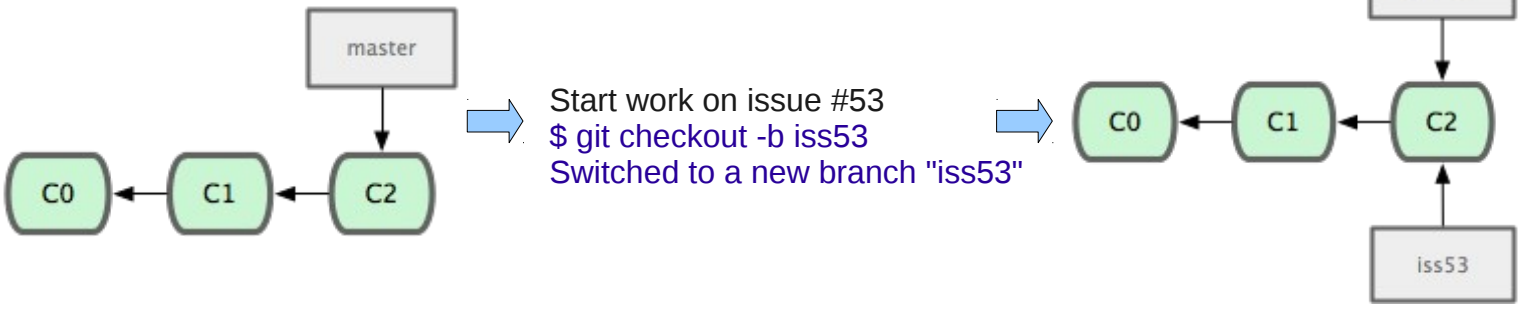


Branch Example

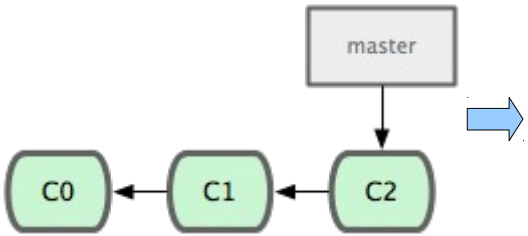


Start work on issue #53
$ git checkout -b iss53
Switched to a new branch "iss53"

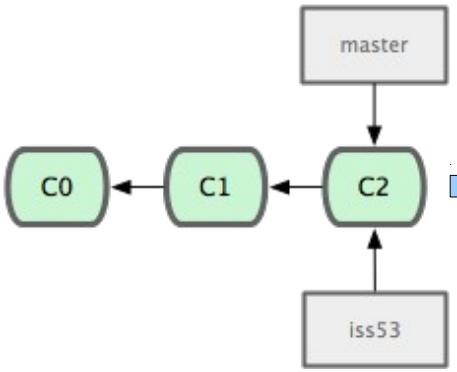
Branch Example



Branch Example

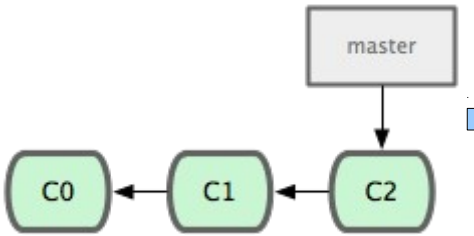


Start work on issue #53
$ git checkout -b iss53
Switched to a new branch "iss53"

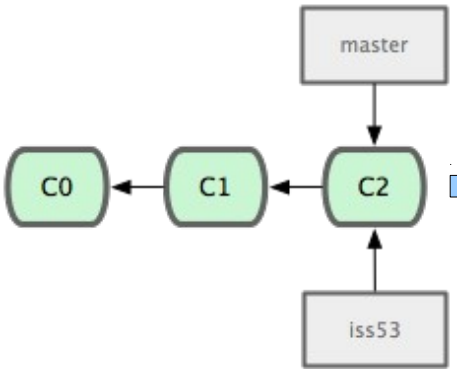


Make changes & commit them
$ git add *modifiedFiles*
$ git commit -m ”describe partial updates for issue 53”

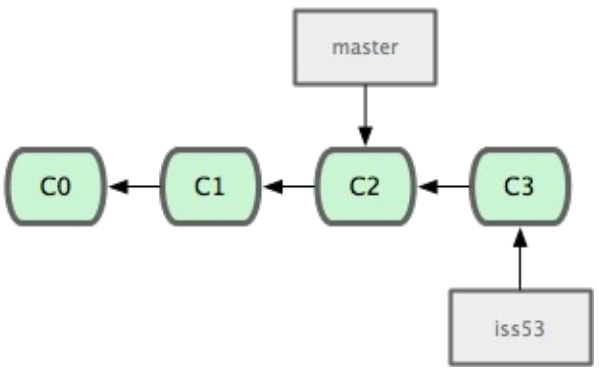
Branch Example



Start work on issue #53
$ git checkout -b iss53
Switched to a new branch "iss53"

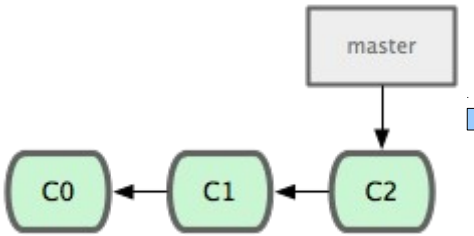


Make changes & commit them
$ git add *modifiedFiles*
$ git commit -m ”describe partial updates for issue 53”

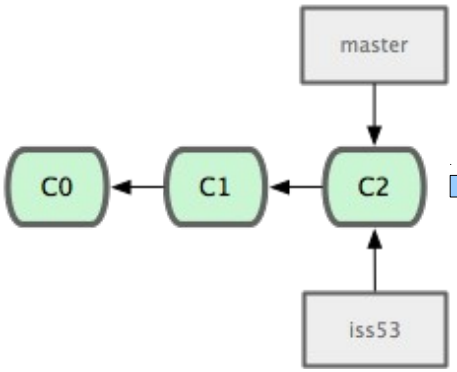


From: http://progit.org/book/ch3-2.html

Branch Example



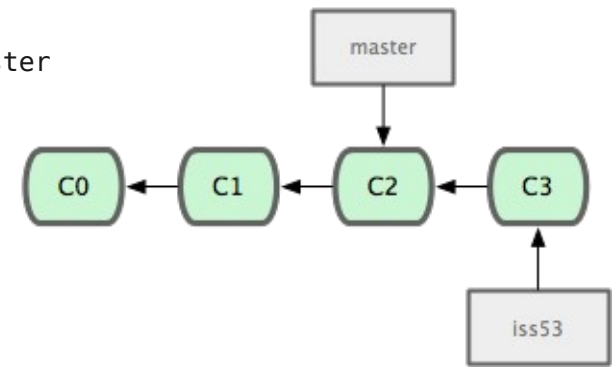
Start work on issue #53
$ git checkout -b iss53
Switched to a new branch "iss53"



Make changes & commit them
$ git add *modifiedFiles*
$ git commit -m ”describe partial updates for issue 53”

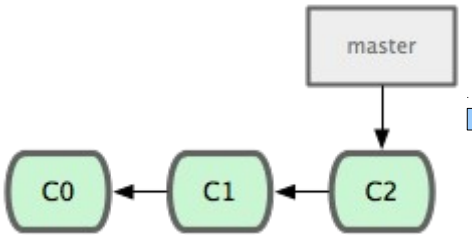
Need to make a different fix based off master

```
$ git checkout master
Switched to branch "master"
$ git checkout -b 'hotfix'
Switched to a new branch "hotfix"
Make changes & commit them
$ git add modifiedFiles
$ git commit -m 'fixed major issue'
```

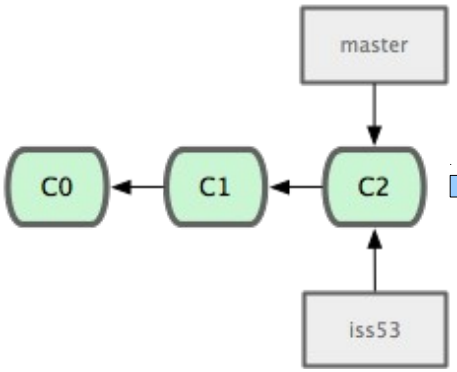


From: http://progit.org/book/ch3-2.html

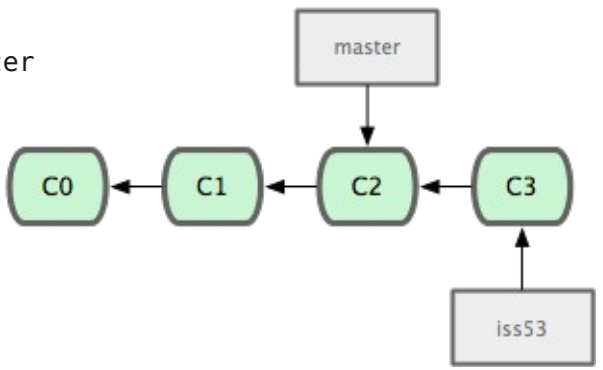
Branch Example



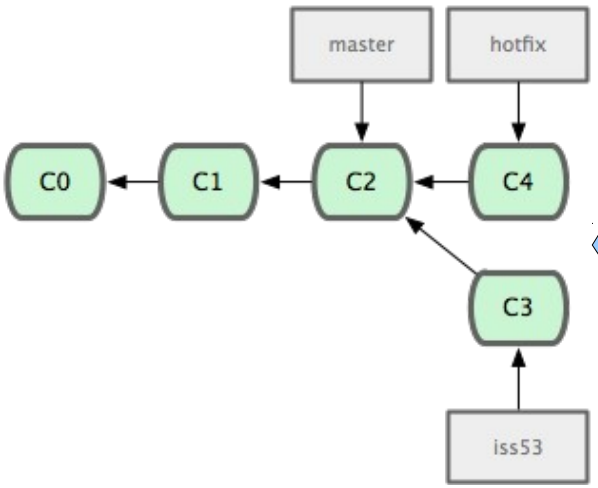
Start work on issue #53
$ git checkout -b iss53
Switched to a new branch "iss53"



Make changes & commit them
$ git add *modifiedFiles*
$ git commit -m ”describe partial updates for issue 53”

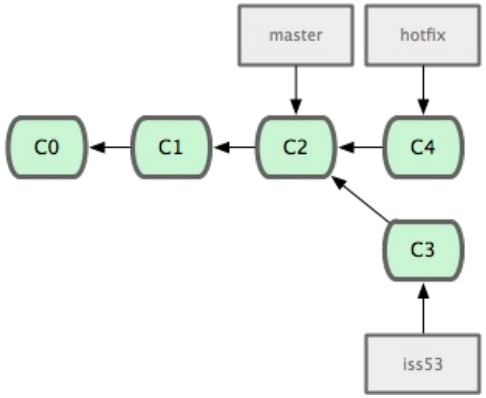



```
Need to make a different fix based off master
$ git checkout master
Switched to branch "master"
$ git checkout -b 'hotfix'
Switched to a new branch "hotfix"
Make changes & commit them
$ git add modifiedFiles
$ git commit -m 'fixed major issue'
```

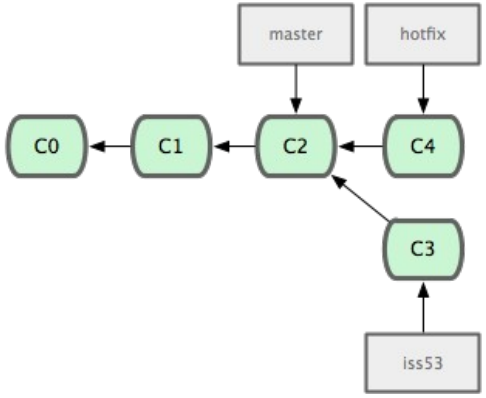


From: http://progit.org/book/ch3-2.html

Merge Example

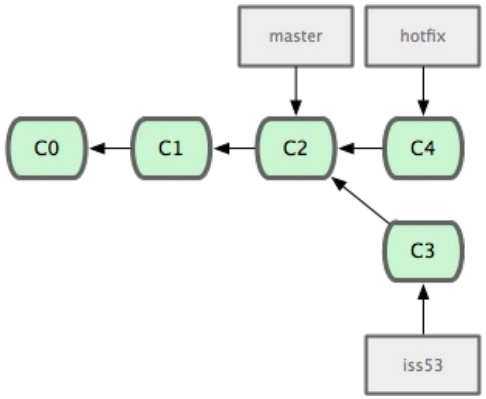


Merge Example

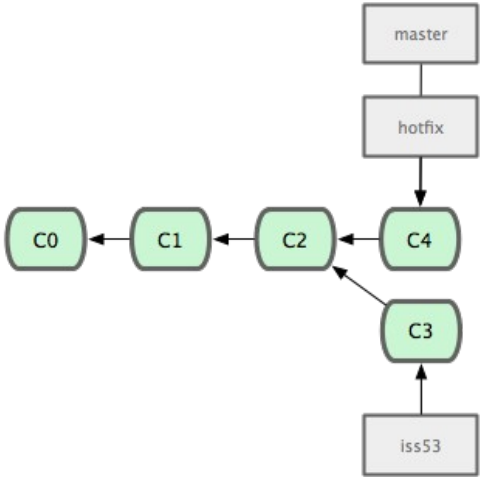



```
Merge hotfix to master
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast forward
...
```

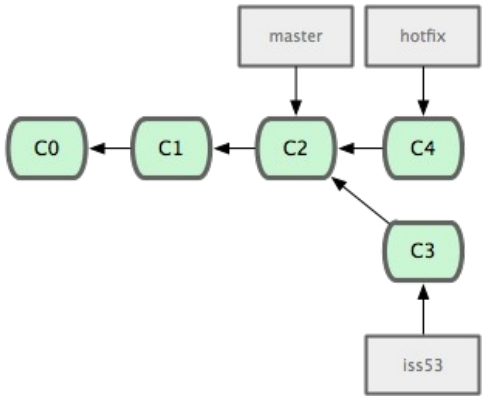
Merge Example



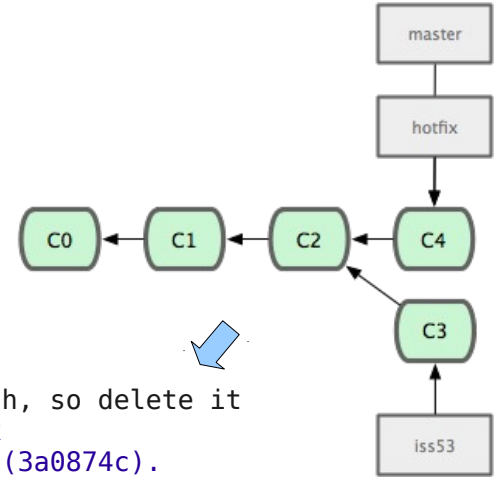

```
Merge hotfix to master
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast forward
...
```



Merge Example

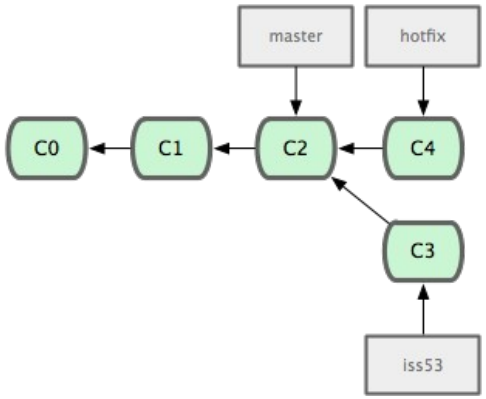



```
Merge hotfix to master
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast forward
...
```

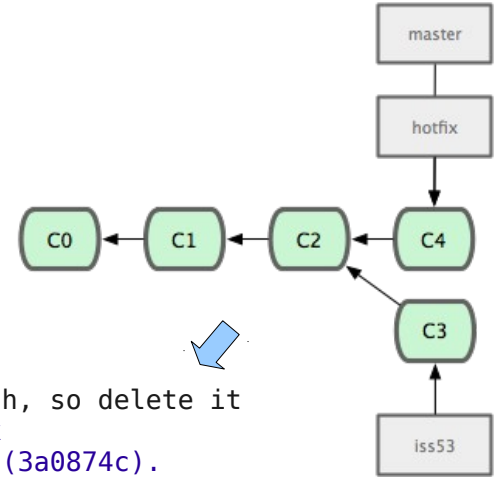



```
Done with hotfix branch, so delete it
$ git branch -d hotfix
Deleted branch hotfix (3a0874c).
Switch back to in-work branch iss53
$ git checkout iss53
Switched to branch "iss53"
Make more changes & commit
$ git add changedFiles
$ git commit -m 'finished description'
```

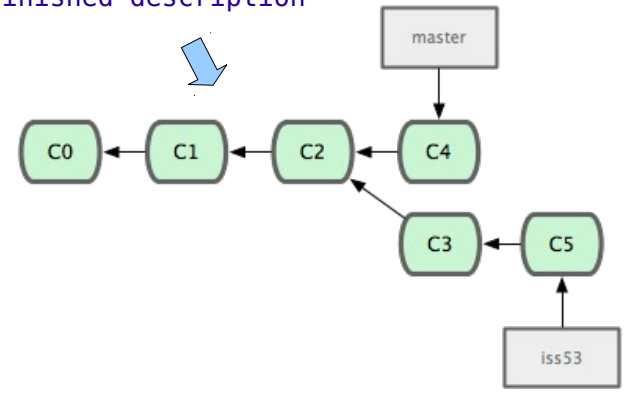
Merge Example



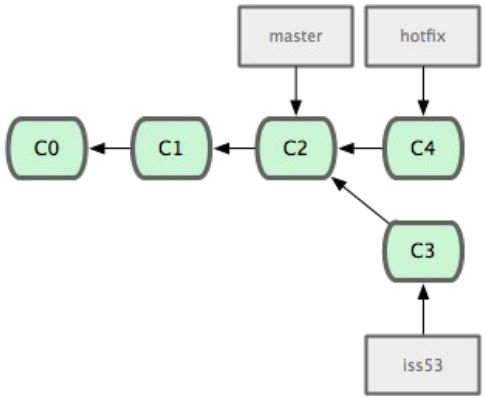

```
Merge hotfix to master
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast forward
...
```



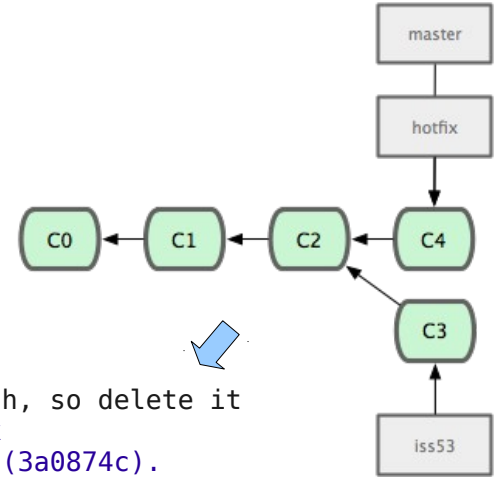

```
Done with hotfix branch, so delete it
$ git branch -d hotfix
Deleted branch hotfix (3a0874c).
Switch back to in-work branch iss53
$ git checkout iss53
Switched to branch "iss53"
Make more changes & commit
$ git add changedFiles
$ git commit -m 'finished description'
```



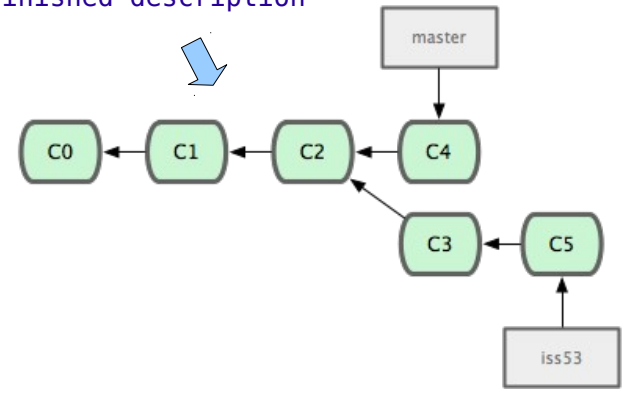
Merge Example




```
Merge hotfix to master
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast forward
...
```

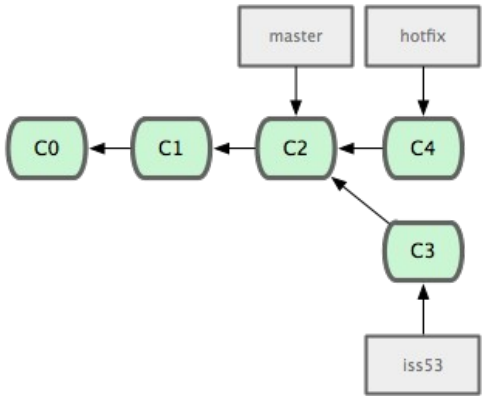



```
Done with hotfix branch, so delete it
$ git branch -d hotfix
Deleted branch hotfix (3a0874c).
Switch back to in-work branch iss53
$ git checkout iss53
Switched to branch "iss53"
Make more changes & commit
$ git add changedFiles
$ git commit -m 'finished description'
```

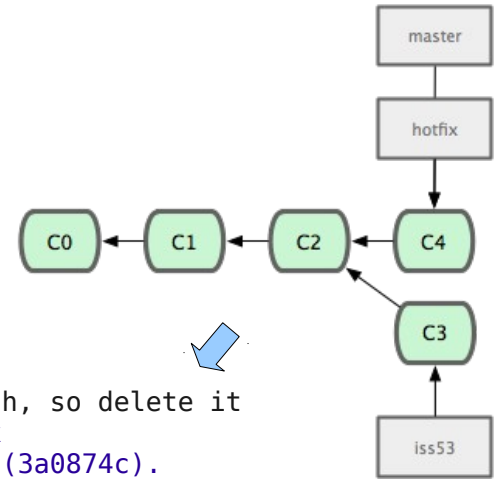



```
Merge into master
$ git checkout master
$ git merge iss53
Merge made by recursive.
...
```

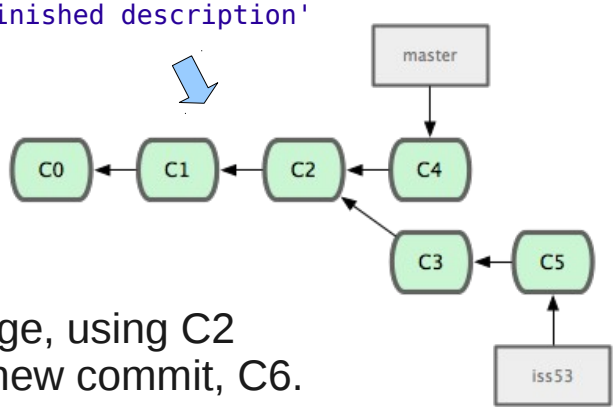
Merge Example



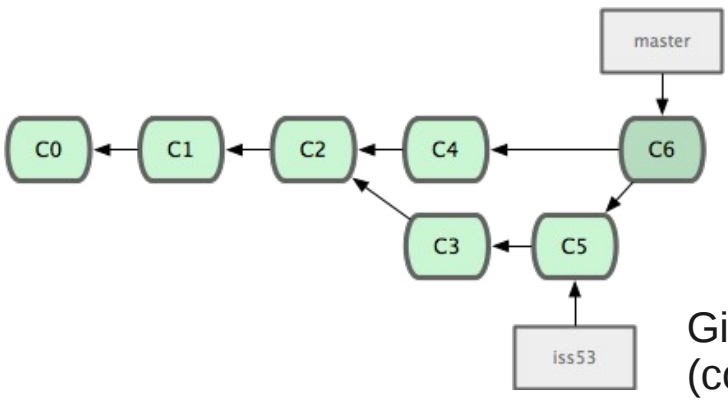

```
Merge hotfix to master
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast forward
...
```




```
Done with hotfix branch, so delete it
$ git branch -d hotfix
Deleted branch hotfix (3a0874c).
Switch back to in-work branch iss53
$ git checkout iss53
Switched to branch "iss53"
Make more changes & commit
$ git add changedFiles
$ git commit -m 'finished description'
```




```
Merge into master
$ git checkout master
$ git merge iss53
Merge made by recursive.
...
```



Git automatically does a 3-way merge, using C2 (common ancestor), C4, & C5 into new commit, C6.

From: http://progit.org/book/ch3-2.html

Merge Conflict

- Same part of the file modified, git needs your help!

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

Merge conflict – lets you resolve

```
$ git status
index.html: needs merge
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       unmerged:   index.html
#
```

Git status indicates unmerged

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

<<<<<<< - start current branch's version

======= - separate's the 2 versions

Conflicts are marked in the file

>>>>>>> - end other branch's version

Resolving Merge Conflict

- Make appropriate changes
- Delete the <<<<<<<, =======, and >>>>>>>
- Add to the staged files
  - git add *nowMergedFile*
- Commit the merge
  - git commit
    - Update the default merge message with a description of how you resolved the merge
- You can also use a mergetool: git mergetool

Sharing Branches

- Like tags not automatically pushed
  - git push origin *branchName* ← Only needed once
- To see all branches including remotes:
  - git branch -a
- Base work on a remote branch/merge back to it
  - git checkout --track origin/*branchName* ← Only needed once
  - Creates & checkouts branch called *branchName*
- Now push & pull from your *branchName*

Cleaning up Branches

- Delete branch after merging & you are done with it
  - git branch -d *branchName*
  - Make sure you have merged first!
- Delete remote branch:
  - git push origin :*branchName*

Resources

- https://statgen.sph.umich.edu/wiki/How_To_Use_Git
- http://progit.org/book/
- http://www.kernel.org/pub/software/scm/git/docs/git.html