Lecture 5
More on Branches and Merging

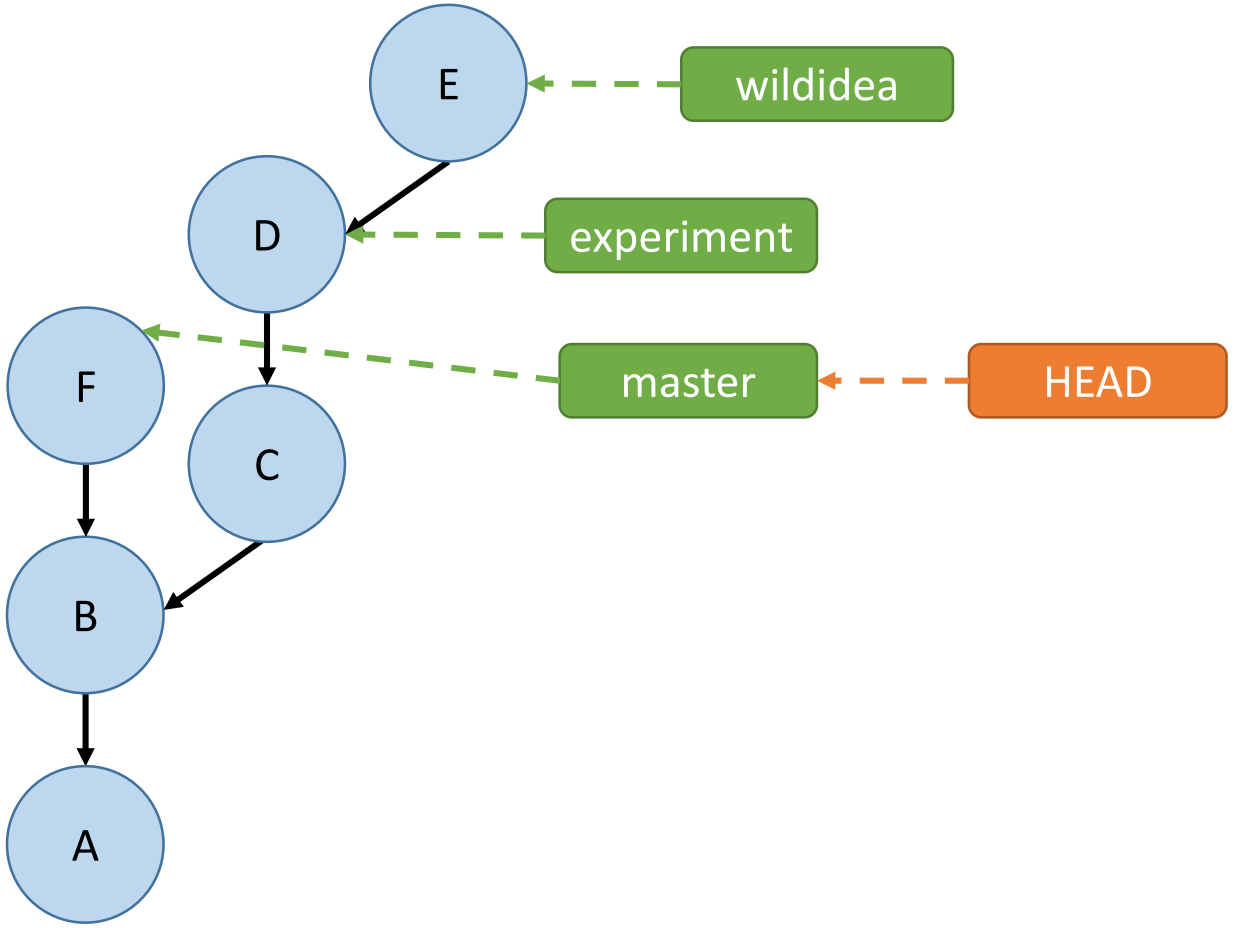
Sign in on the attendance sheet!

Midterm Next Week

- During class
- Very similar to the homework problems
- Don’t stress!
- No homework this week, but practice all of the commands we’ve learned.

Last Time

- Branches are pointers to specific commits
- Branches allow us to create commit histories that diverge
- We can merge diverged histories back together

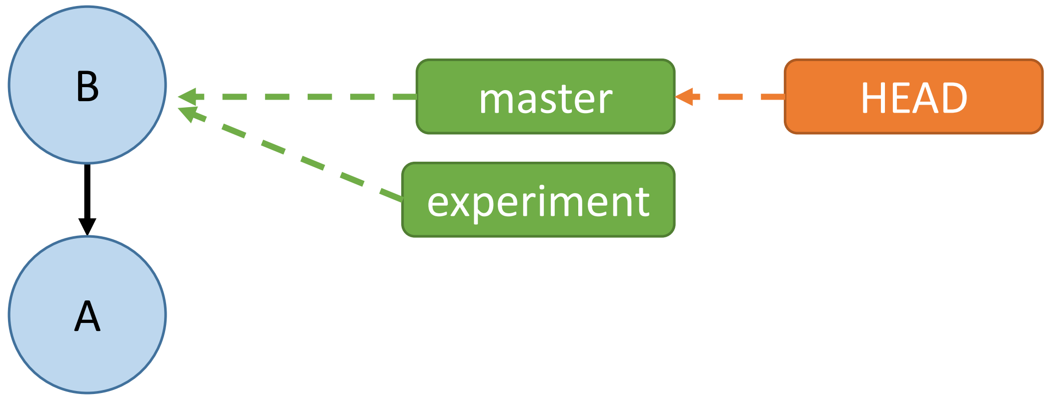


`git branch <newbranchname>`

Example use:

```
git branch experiment
```

- Creates a new branch called “develop” that **points** to wherever you are right now (i.e. wherever HEAD is right now)

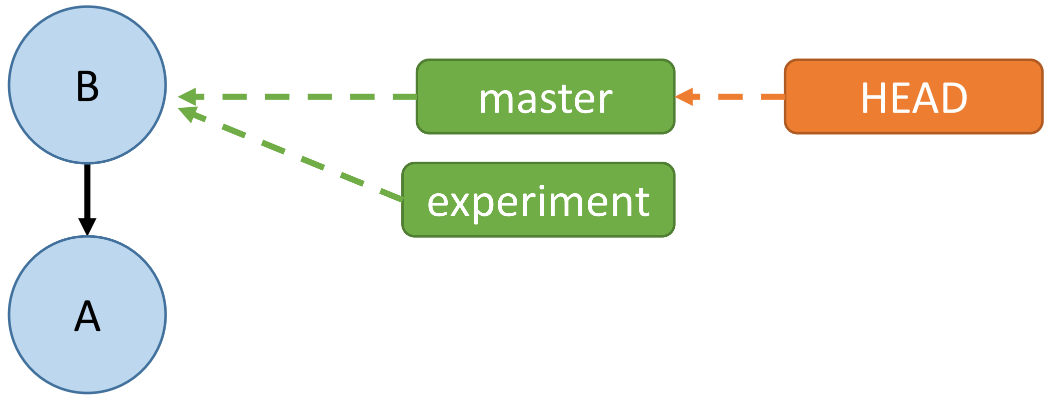


`git checkout <branchname>`

Example use:

```
git checkout experiment
```

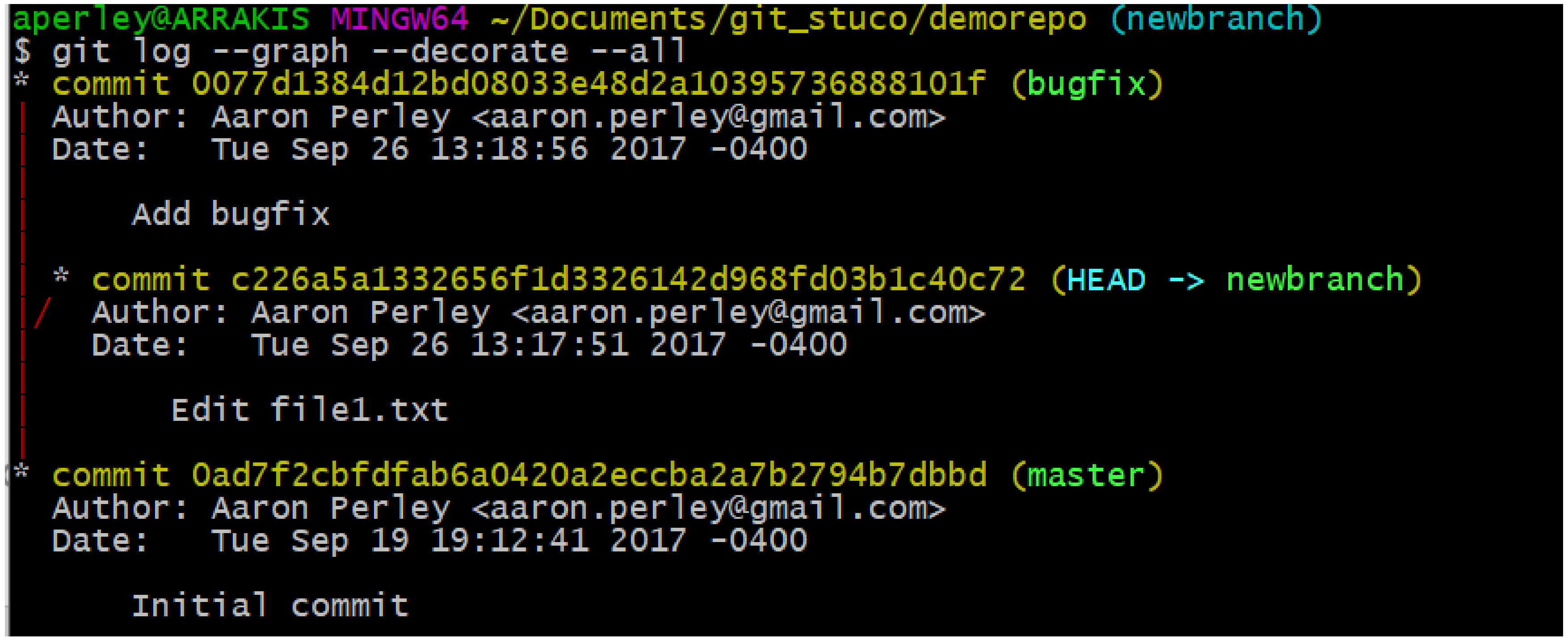
Switches the HEAD to the branch named “develop”



Git Log and Branches

git log does not show all branches well by itself. Use:

`git log --graph --decorate --all`



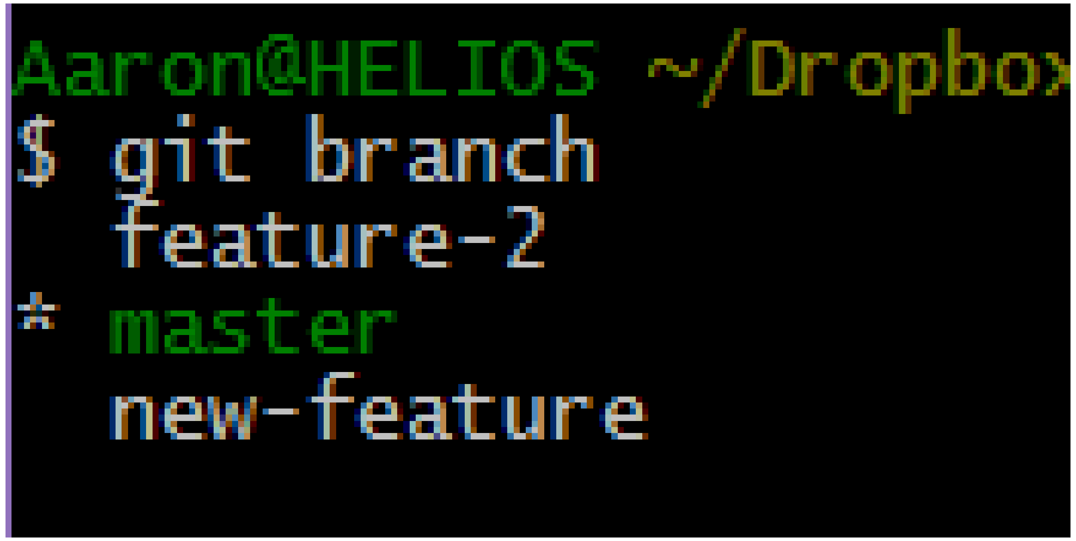
```
aperley@ARRAKIS MINGW64 ~/Documents/git_stuco/demorepo (newbranch)
$ git log --graph --decorate --all
* commit 0077d1384d12bd08033e48d2a10395736888101f (bugfix)
| Author: Aaron Perley <aaron.perley@gmail.com>
| Date:   Tue Sep 26 13:18:56 2017 -0400
|
|     Add bugfix
|
| * commit c226a5a1332656f1d3326142d968fd03b1c40c72 (HEAD -> newbranch)
|/  Author: Aaron Perley <aaron.perley@gmail.com>
|   Date:   Tue Sep 26 13:17:51 2017 -0400
|
|       Edit file1.txt
|
* commit 0ad7f2cbfdfab6a0420a2eccba2a7b2794b7dbbd (master)
  Author: Aaron Perley <aaron.perley@gmail.com>
  Date:   Tue Sep 19 19:12:41 2017 -0400

      Initial commit
```

git branch

Example use:

`git branch`



```
Aaron@HELIOS ~/Dropbo
$ git branch
  feature-2
* master
  new-feature
```

- Lists all the local branches in the current repository and marks which branch you’re currently on
  - Where are “you”? Well, you’re always at HEAD. Usually, you’re also at a branch as well.
- The default branch in a repository is called “master”

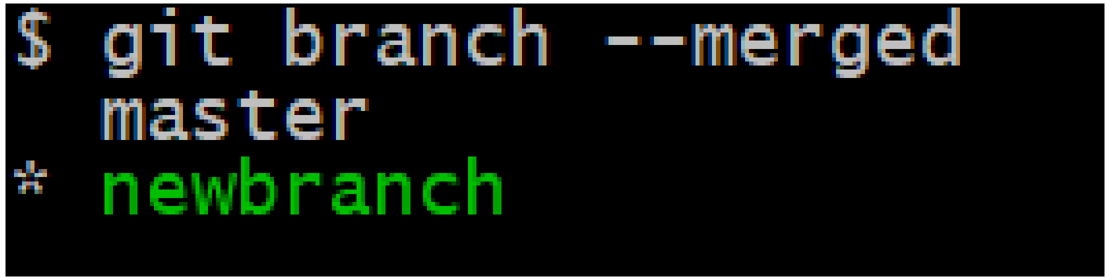
Telling if branches are merged

`git branch --merged`

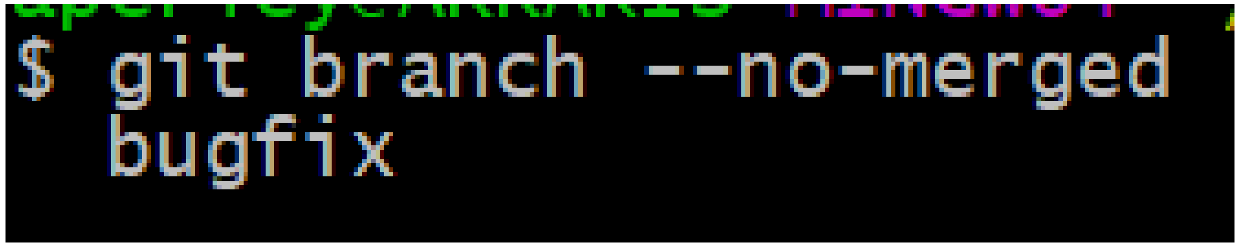
Lists all branches merged into the current branch

`git branch --no-merged`

Lists all unmerged branches relative to the current branch



```
$ git branch --merged
  master
* newbranch
```



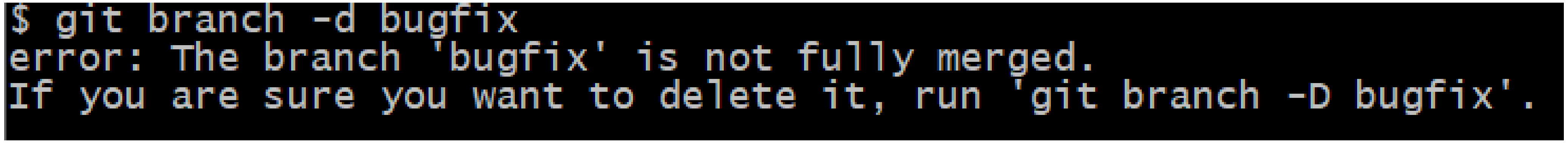
```
$ git branch --no-merged
  bugfix
```

A branch is merged into the current branch if it is an ancestor of that branch!

Deleting Branches

`git branch -d <branchname>`

Will only delete the branch if it is merged into HEAD



```
$ git branch -d bugfix
error: The branch 'bugfix' is not fully merged.
If you are sure you want to delete it, run 'git branch -D bugfix'.
```

`git branch -D <branchname>`

Will force delete the branch

Deleting a branch just removes the pointer!

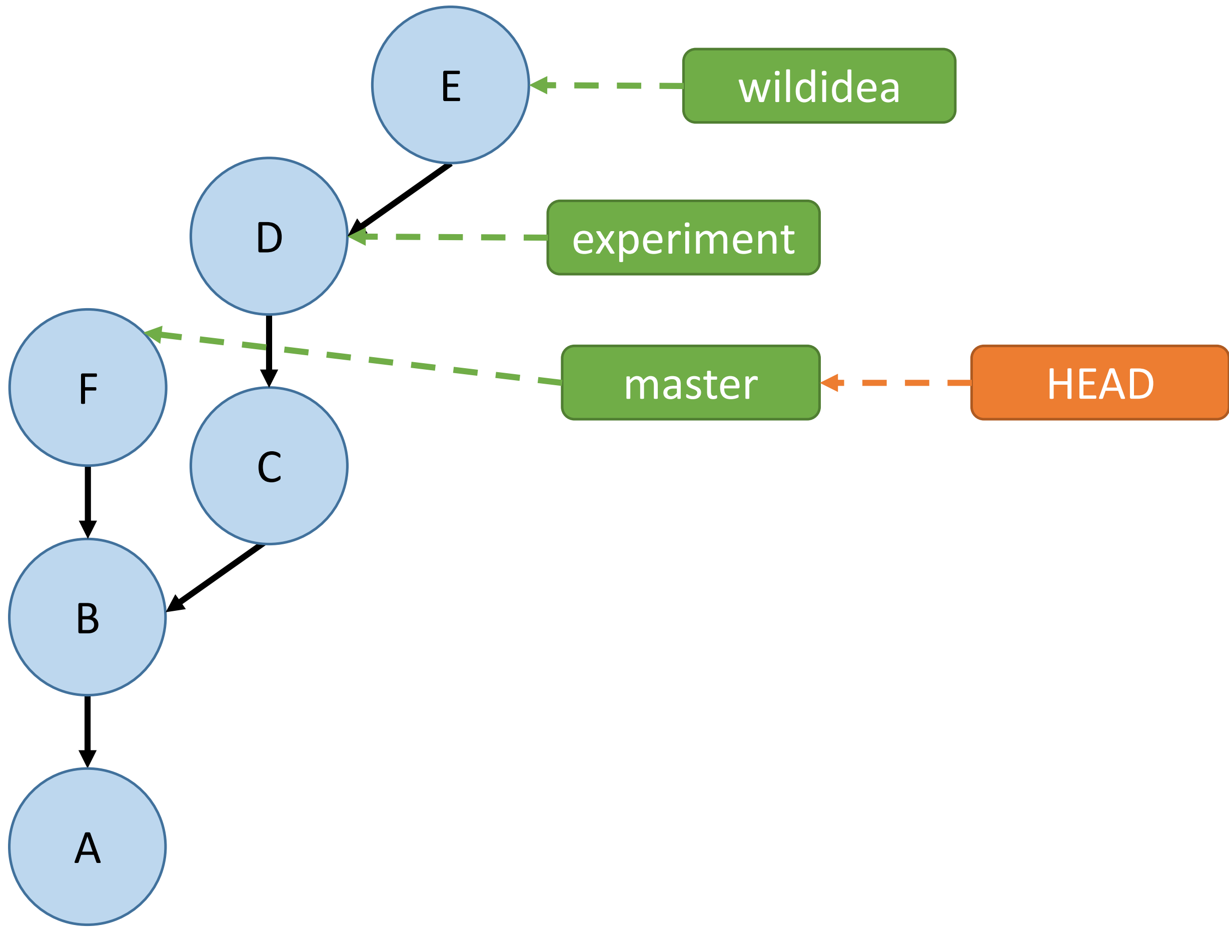
Git Diff with Branches

```
git diff branch1…branch2
```

View changes on branch2, starting at the common ancestor of branch1 and branch2.

```
$ git diff master…experiment
```

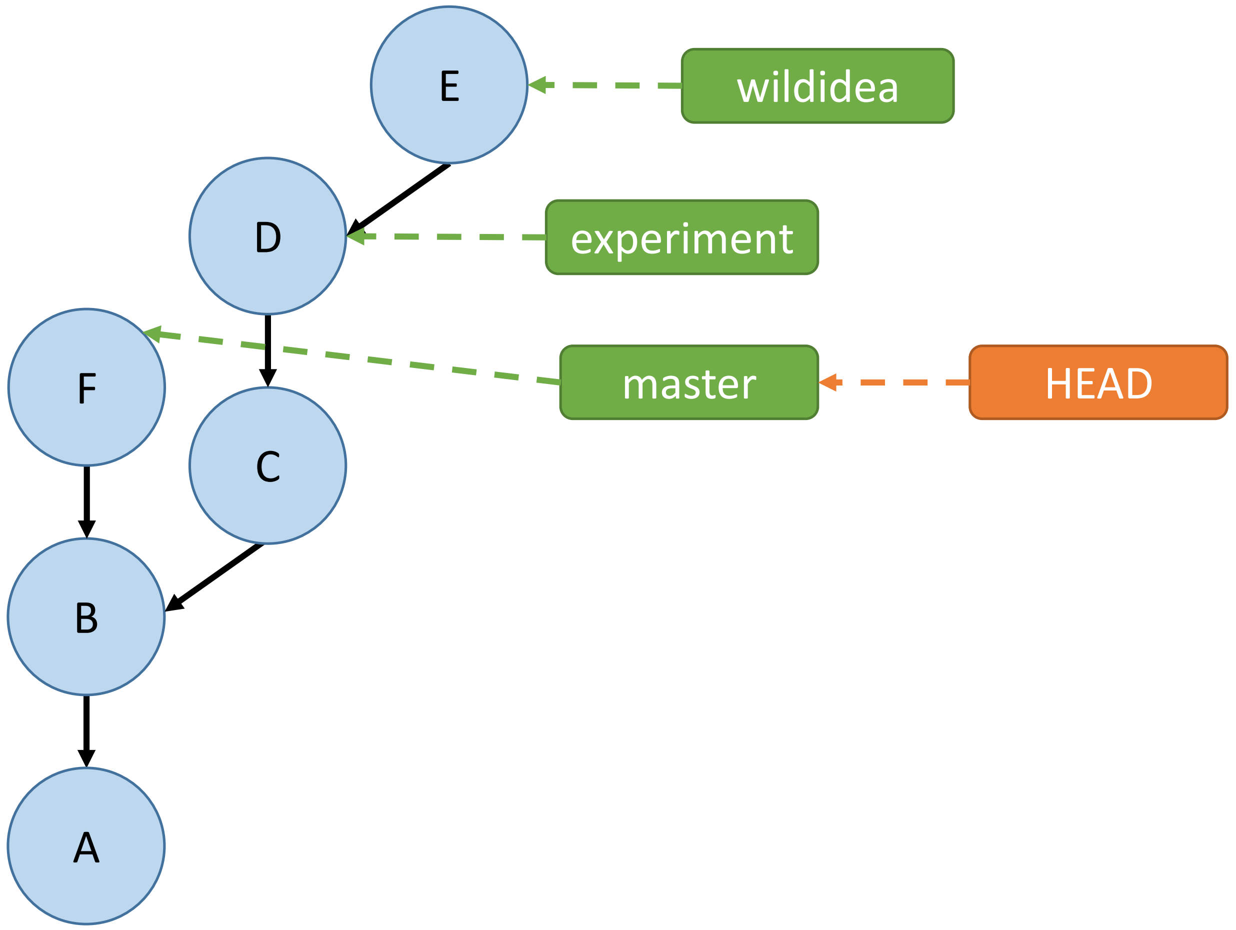
Shows changes in commits C and D, but not F



Merging

```
git merge experiment
```

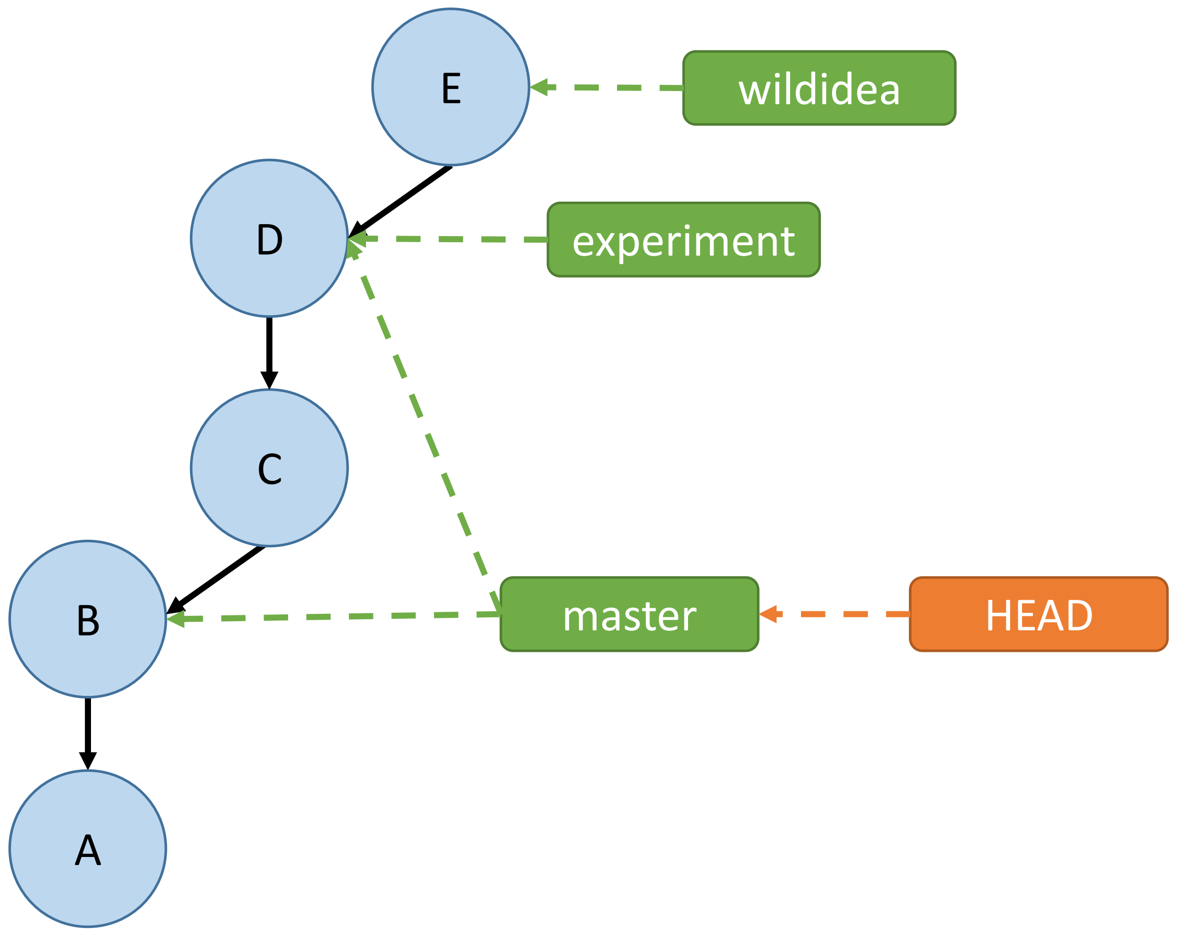
"Will replay the changes made on the experiment branch since it diverged from master (i.e. B) until its current commit (D) on top of master, and record the result in a new commit along with the names of the two parent commits." (from git help merge)



Fast Forward Merges

- Occur when the branch being merged **onto** is an **ancestor** of the branch being **in**.
- No merge commit is made unless --no-ff flag is used
- Will never cause conflicts!

```
git merge experiment
```

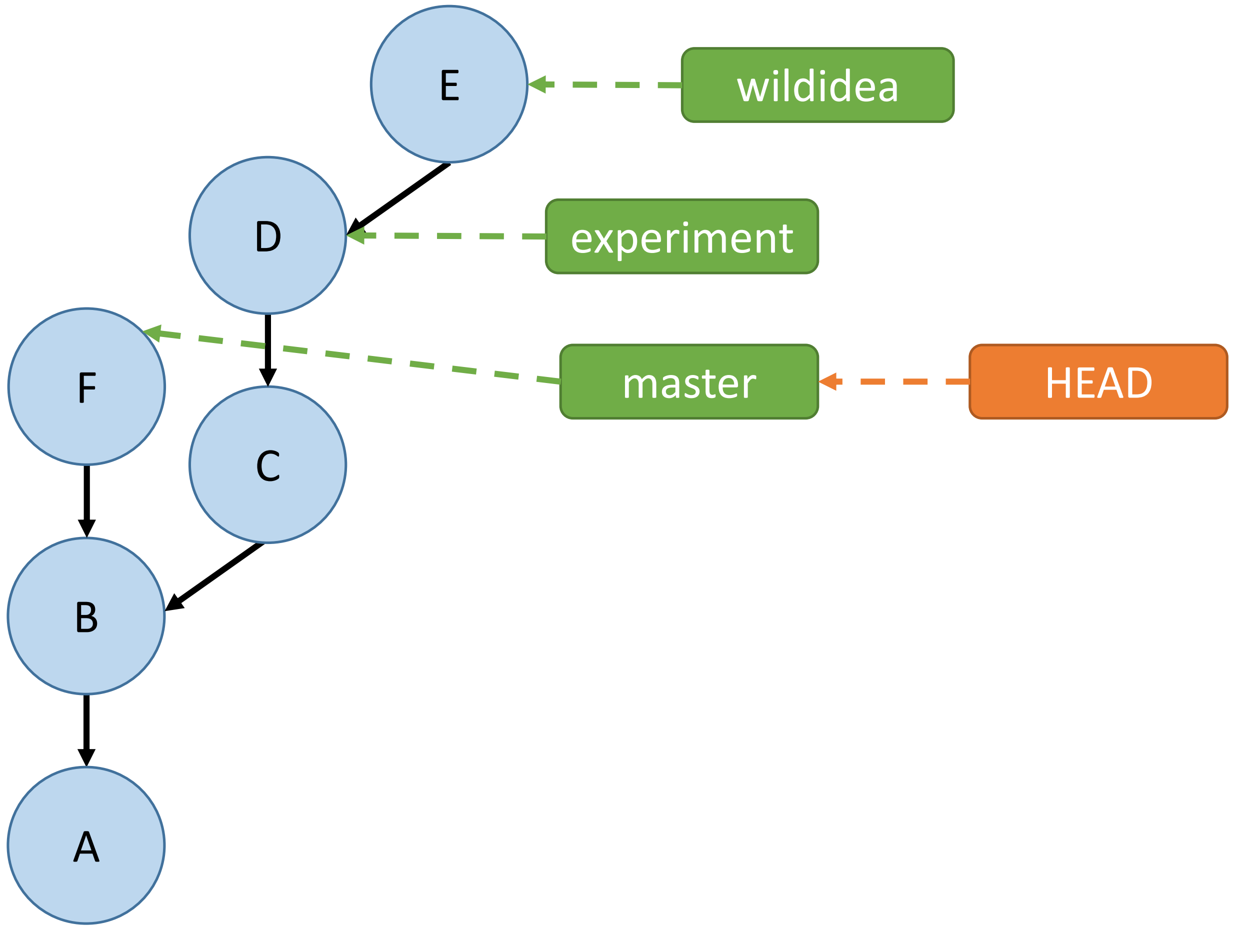


Three-Way Merges

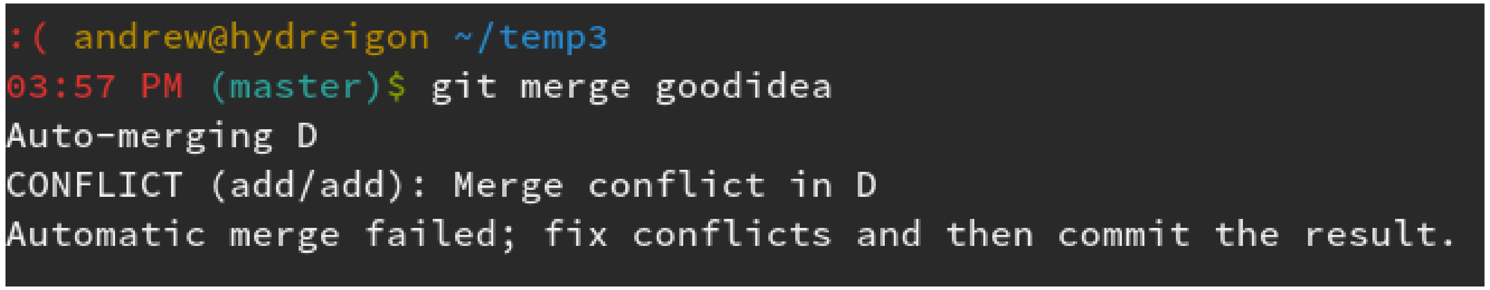
- Occur when the branch being merged **onto** is **not a descendent** of the branch being merged **in**.
- The branch being merged onto has "moved on" since the split.
- Creates a merge commit, can cause conflicts!

```
git merge experiment
```

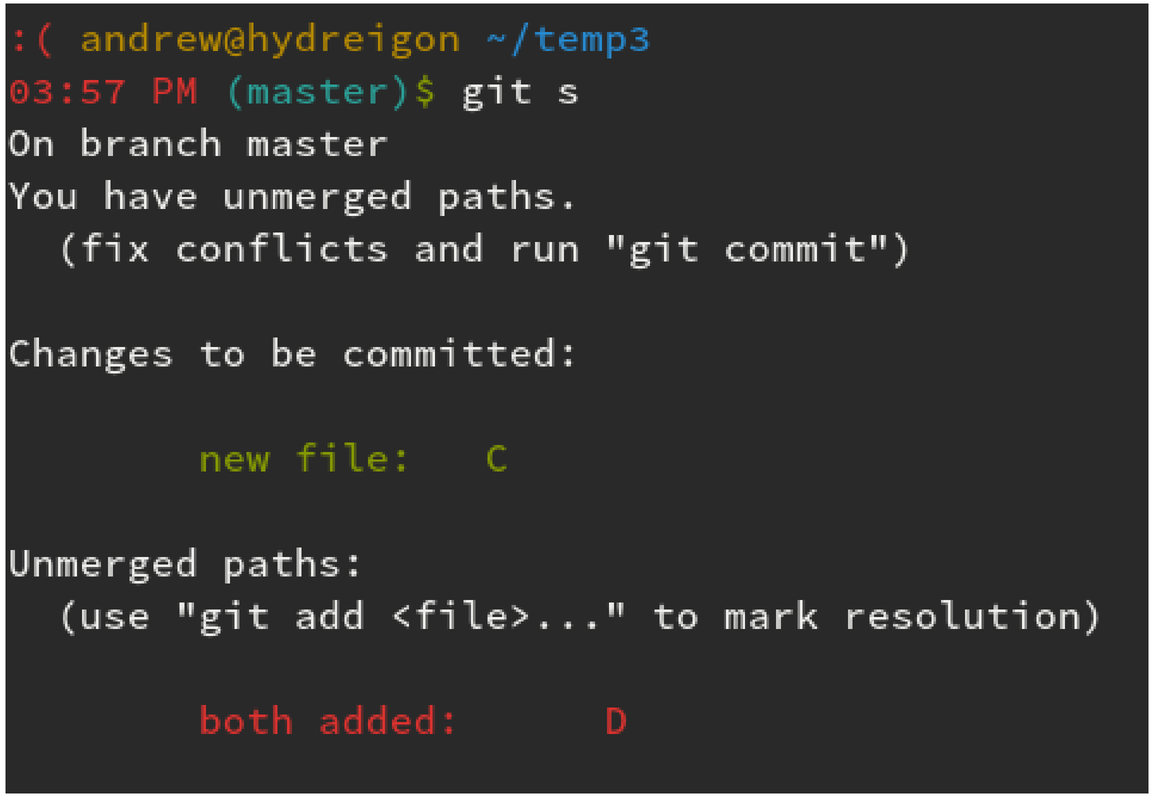
Performs a "three way merge" between B, F, and D



MERGE CONFLICT



```
:( andrew@hydreigon ~/temp3
03:57 PM (master)$ git merge goodidea
Auto-merging D
CONFLICT (add/add): Merge conflict in D
Automatic merge failed; fix conflicts and then commit the result.
```



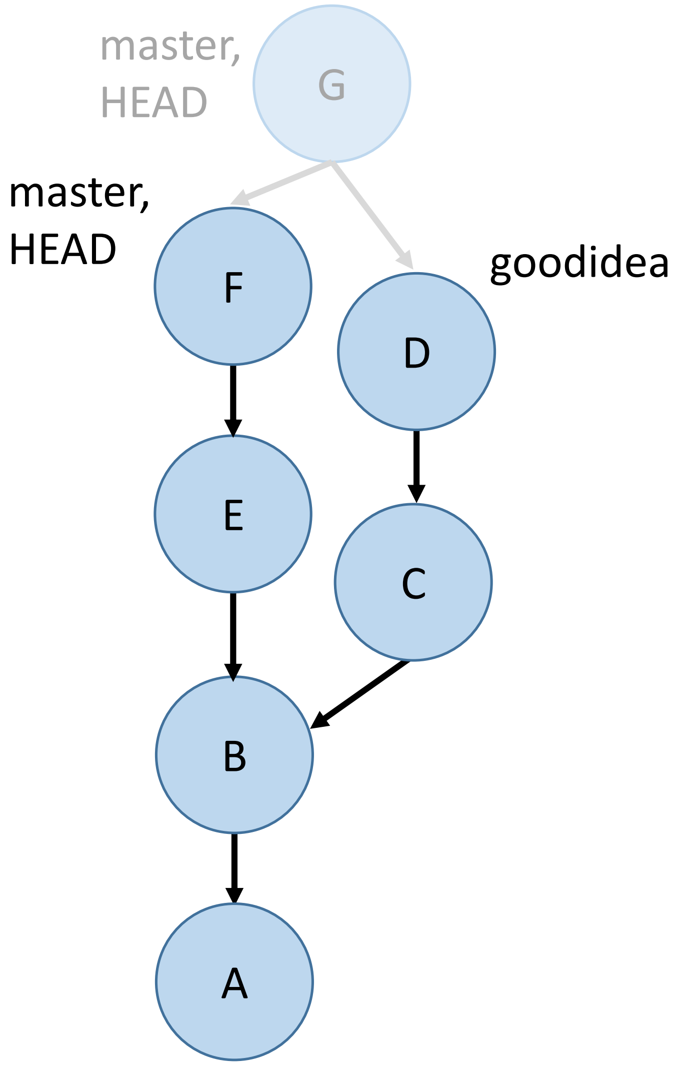
```
:( andrew@hydreigon ~/temp3
03:57 PM (master)$ git s
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Changes to be committed:

        new file:   C

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both added:      D
```



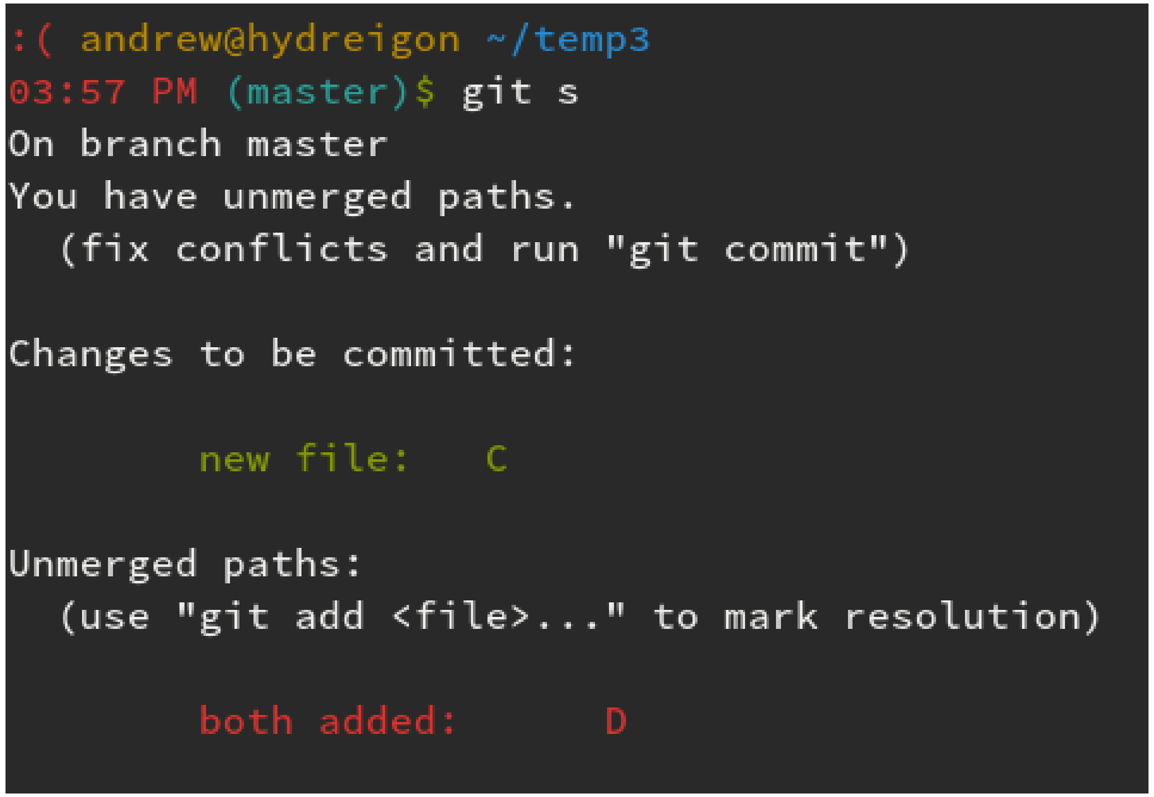
MERGE CONFLICT

```
This file is demo.txt

<<<<<<< HEAD
Here is another line. modified in master
=======
Here is another line. modified in goodidea
>>>>>>> goodidea
```

“How to fix a merge conflict”

- Run `git status` to find the files that are in conflict.
- For each of these files, look for lines like “<<<<<<< HEAD” or “>>>>>>> 3de67ca” that indicate a conflict.
- Edit the lines to match what you want them to be.
- After you finish doing this for each conflict in each file, `git add` these conflicted files and run `git commit` to complete the merge.



```
:( andrew@hydreigon ~/temp3
03:57 PM (master)$ git s
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Changes to be committed:

        new file:   C

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both added:      D
```

Activity!

1. Create a git repository
2. Make a new file called file1.txt, add some lines to it, and commit it
3. Create two branches called branch1 and branch2
4. Edit the same line in the text file and make a commit in each branch
5. Switch computers with the person next to you and try to merge both branches back to master (merging the second branch back will require resolving the conflicts).
6. What do we call the merge that occurred when merging the first branch back to master?