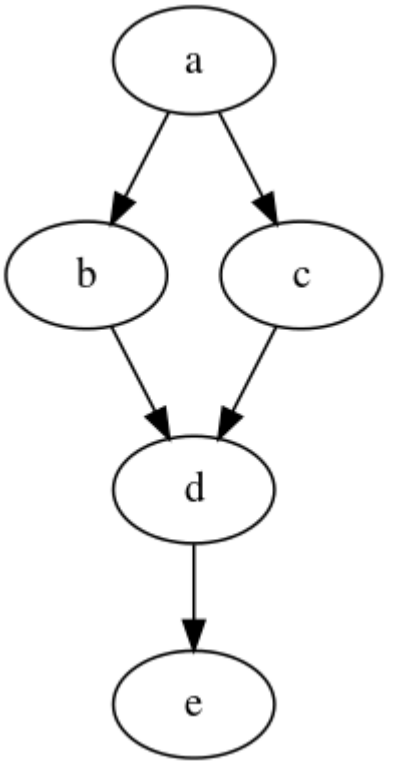
Git: Merge vs. Rebase

A brownbag workshop at

by Seth House

@whiteinge
seth@eseth.com

Git Mental Model



Directed acyclic graph (DAG).

Branches are pointers to nodes

```
% cat .git/refs/heads/master
6b64c2889d97b456f4682fdbe8c46d69e556c24d
```

Branches are pointers to nodes

```
% cat .git/refs/heads/master
6b64c2889d97b456f4682fdbe8c46d69e556c24d
```

See also: `.git/packed-refs`.

Destructive Operations

You don't remove data from Git.
Git garbage collects data itself.

Destructive Operations

You don't remove data from Git.
Git garbage collects data itself.

If it's in a commit, it's safe.

Destructive Operations

You don't remove data from Git.
Git garbage collects data itself.

If it's in a commit, it's safe.

After 90 days.
`git reflog --date=relative`

Utils

~/.gitconfig

```
[alias]

# Make a stub commit with file and file contents.
stub = "!sh -c 'echo $1 > $1; git add $1; git commit -m Add-$1' -"
```

`git-graph-dag`

1. https://github.com/whiteinge/dotfiles/blob/master/bin/git-graph-dag
2. Add to your `$PATH`.
3. Available as `git graph-dag`.
4. See comments for prerequisites and usage.

`git-graph-dag`

1. https://github.com/whiteinge/dotfiles/blob/master/bin/git-graph-dag
2. Add to your `$PATH`.
3. Available as `git graph-dag`.
4. See comments for prerequisites and usage.

```
% git graph-dag -c [...<refA> <refB>] \
    | dot -Tpng \
    | open -a Preview.app -f
```

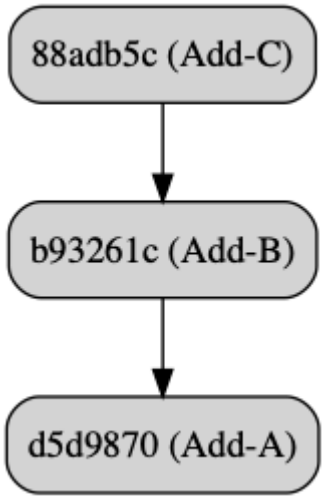
Set up

```
% git init mergerebase
Initialized empty Git repository in /private/tmp/mergerebase/.git/

% cd mergerebase
/private/tmp/mergerebase
```

Merge

Commit to `master`

```
% git stub A
[master (root-commit) d5d9870] A
 1 file changed, 1 insertion(+)
 create mode 100644 A

% git stub B
[master b93261c] Add-B
 1 file changed, 1 insertion(+)
 create mode 100644 B

% git stub C
[master 88adb5c] Add-C
 1 file changed, 1 insertion(+)
 create mode 100644 C
```

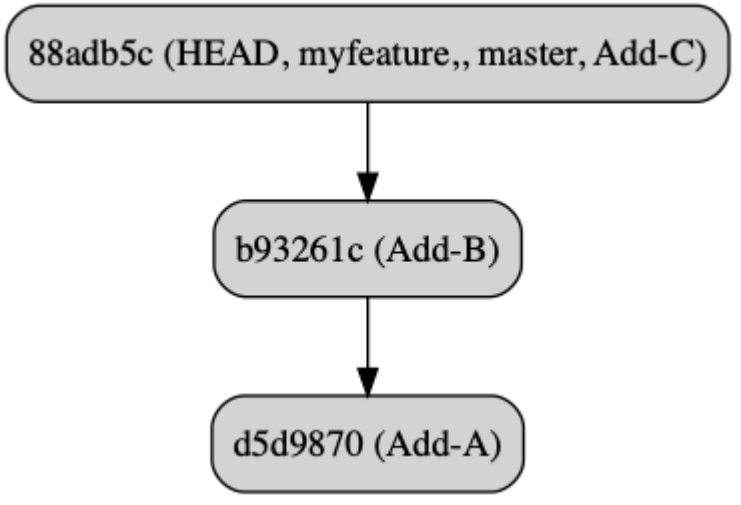


`git graph-dag -c master`

Create `myfeature`

```
% git co -b myfeature
Switched to a new branch 'myfea

% git branch --set-upstream-to=
Branch 'myfeature' set up to tr
```

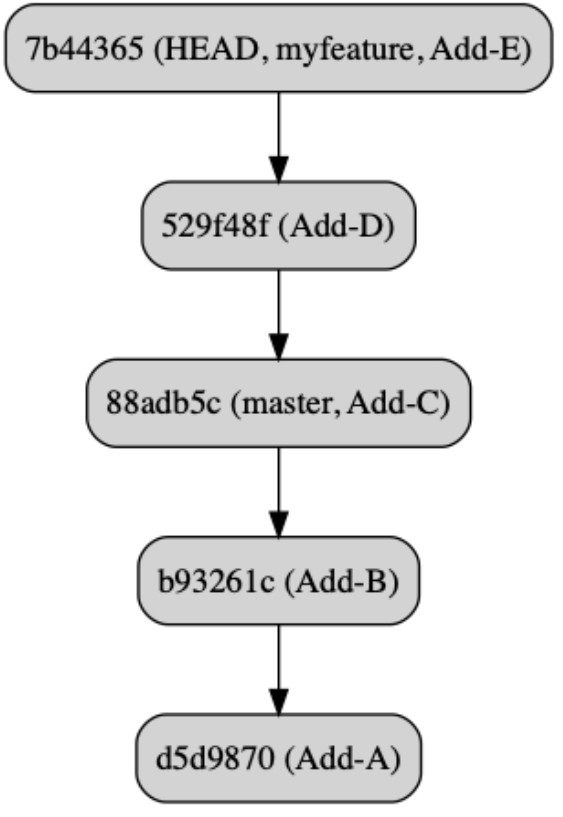
Same commit in the DAG.
Just a new pointer.



`git graph-dag -c myfeature`

Commit to `myfeature`

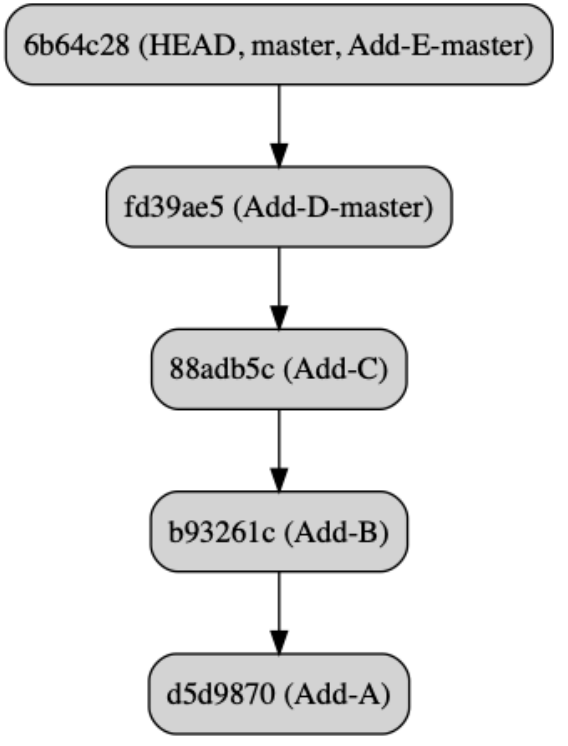
```
% git stub D
[myfeature 529f48f] Add-D
 1 file changed, 1 insertion(+)
 create mode 100644 D

% git stub E
[myfeature 7b44365] Add-E
 1 file changed, 1 insertion(+)
 create mode 100644 E
```



`git graph-dag -c myfeature`

Meanwhile, back on `master`...

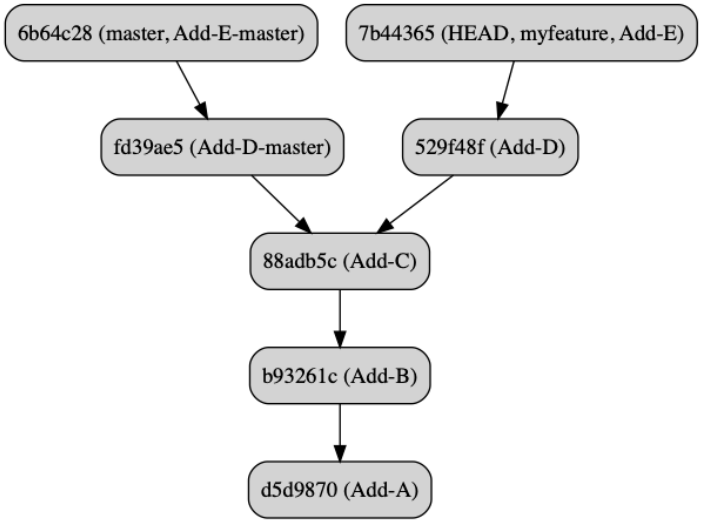
```
% git co -
Switched to branch 'master'

% git stub D-master
[master fd39ae5] Add-D-master
 1 file changed, 1 insertion(+)
 create mode 100644 D-master

% git stub E-master
[master 6b64c28] Add-E-master
 1 file changed, 1 insertion(+)
 create mode 100644 E-master
```



`git graph-dag -c master`

`myfeature` and `master` have diverged

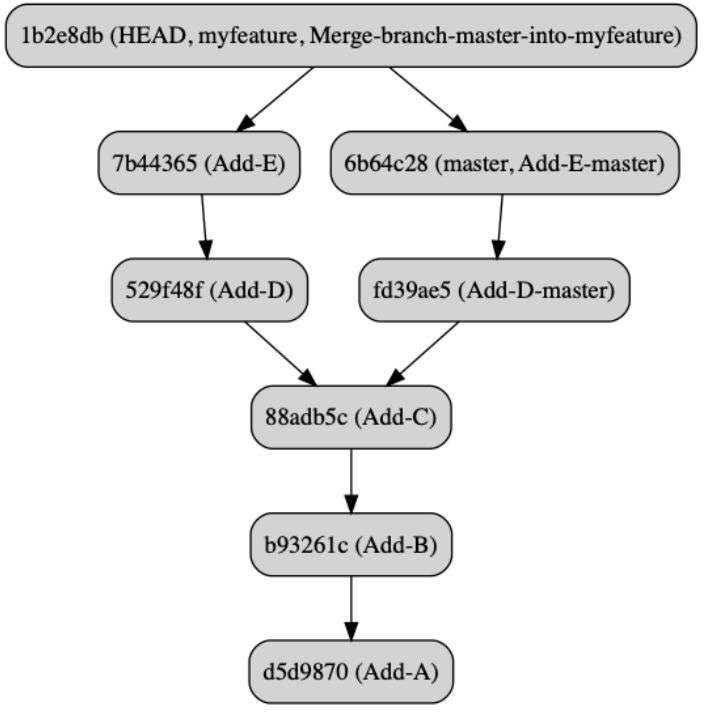
```
% git co -
Switched to branch 'myfeature'
Your branch and 'master' have d
and have 2 and 2 different comm
  (use "git pull" to merge the
```



`git graph-dag -c master myfeature`

Update `myfeature` with `master` via merge

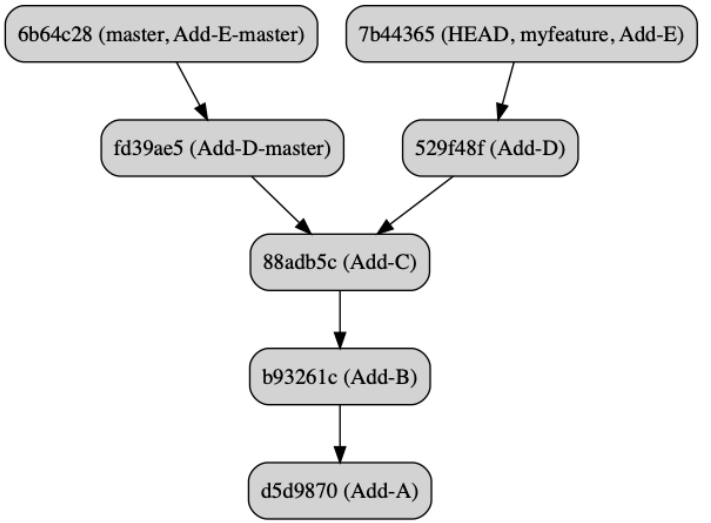
```
% git merge master
Merge made by the 'recursive' s
 D-master | 1 +
 E-master | 1 +
 2 files changed, 2 insertions(
 create mode 100644 D-master
 create mode 100644 E-master

% git rev-parse --short HEAD
1b2e8db
```



`git graph-dag -c master myfeature`

Undo the merge (move the `myfeature` pointer)

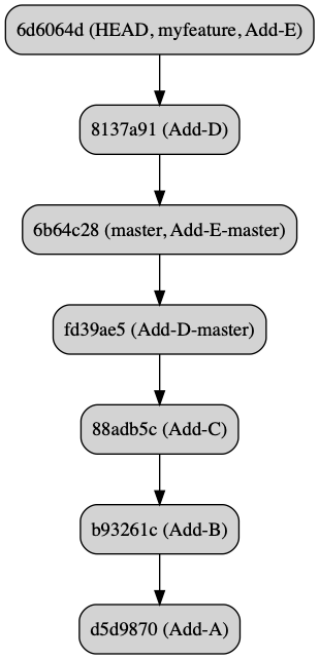
```
% git reset --hard HEAD~1
HEAD is now at 7b44365 Add-E
```



`git graph-dag -c master myfeature`

Update `myfeature` with `master` via rebase

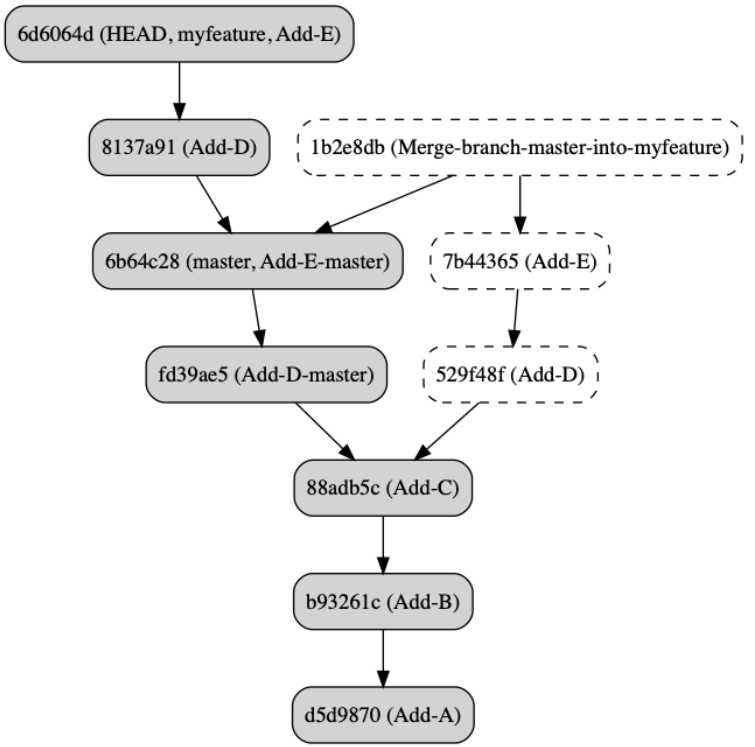
```
% git rebase
First, rewinding head to replay
Applying: Add-D
Applying: Add-E
```



`git graph-dag -c master myfeature`

Both merge and rebase

The merge is still safely in the DAG.




```
git graph-dag -c master myfeature 1b2e8db
```