Git:
Fetch, Pull
Rebase, Merge

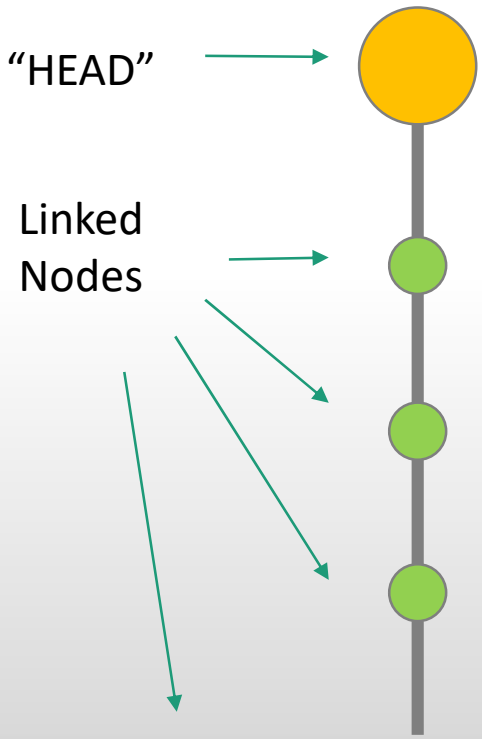
Lauren Billington

Legislative Information Services

Colorado State Legislature

Part 1: Rebase vs Merge

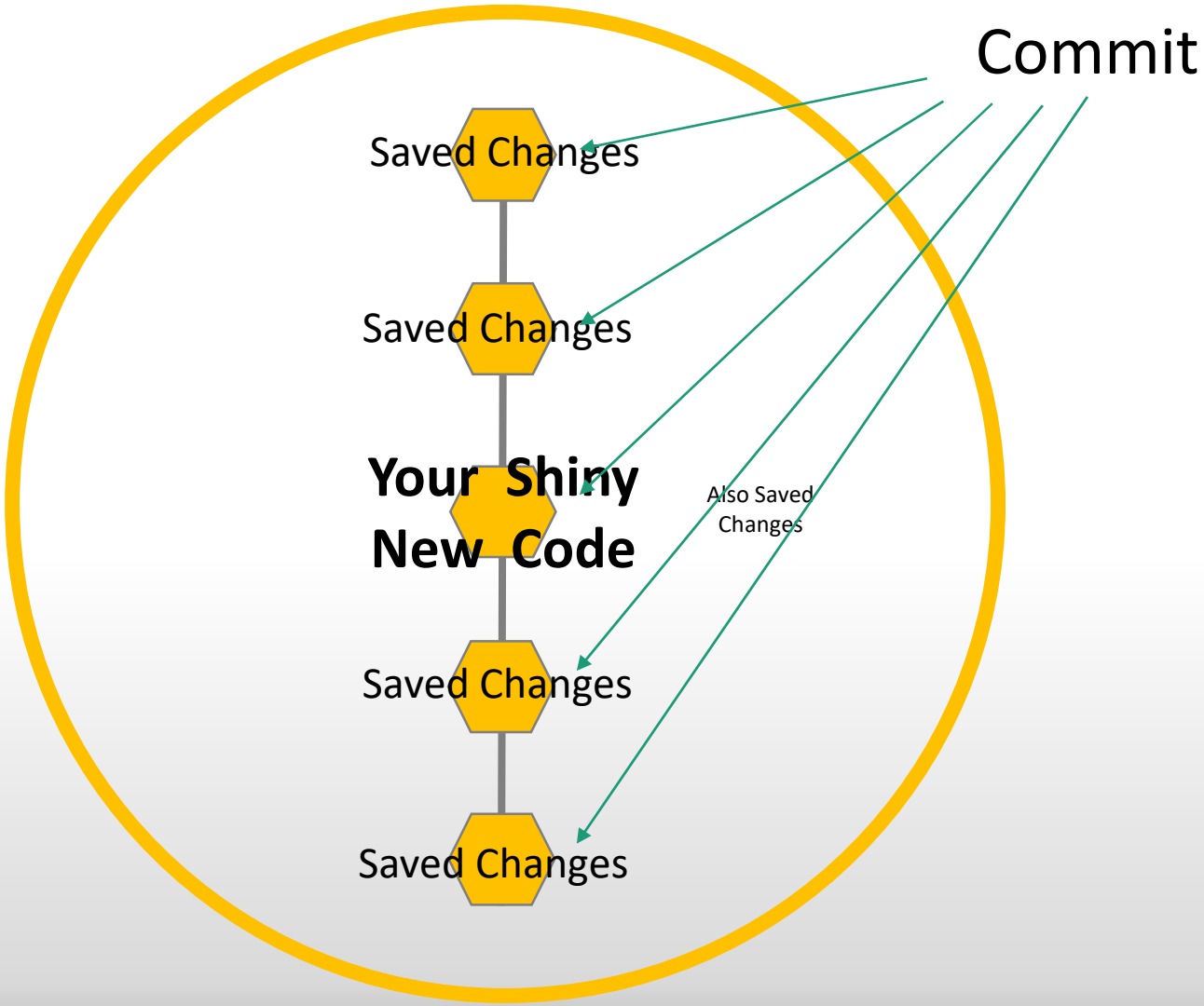
The Classic Visualization

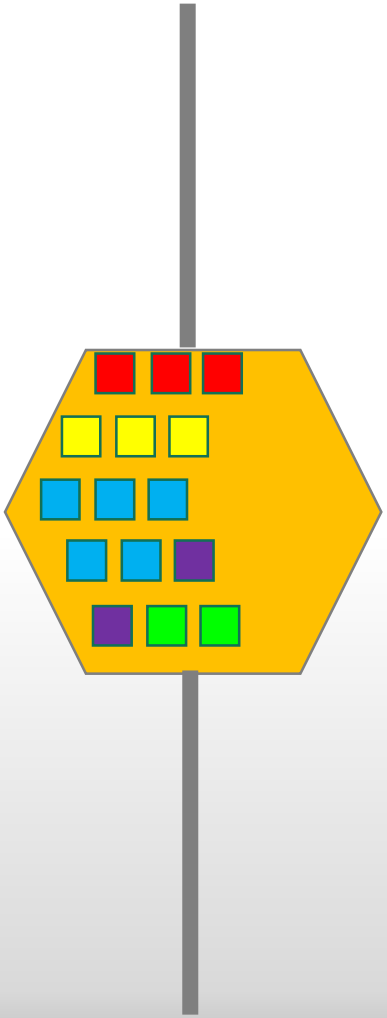


Git Version Control
is basically a
Singly Linked List

The most up-to-date Dot is
always named “Head”

Your code
is actually a Linked List

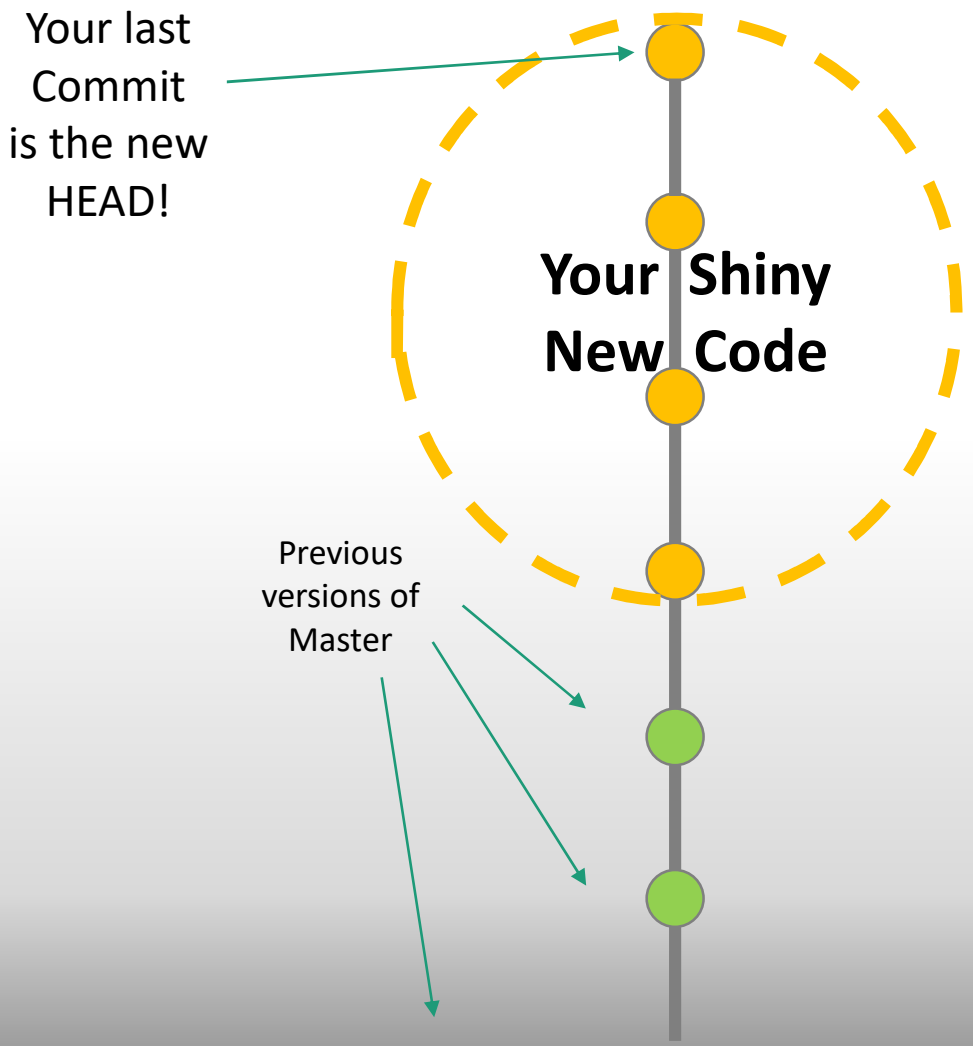

Commit
Saved Changes
Saved Changes
Your Shiny New Code
Also Saved Changes
Saved Changes
Saved Changes



1 Commit **=**

1 copy of the whole codebase
including your new changes

The simplest use case



Rebasing vs Merge

- Rebasing
  - Forces us to treat integration like a Singly Linked List

Rebasing vs Merge

- Merging
  - Attempts one of 6 merging strategies:
    - Fast Forward
    - Recursive
    - Ours
    - Octopus
    - Resolve
    - Subtree

‘git merge’ing Your Code

running ‘git merge’ all by itself could result in…

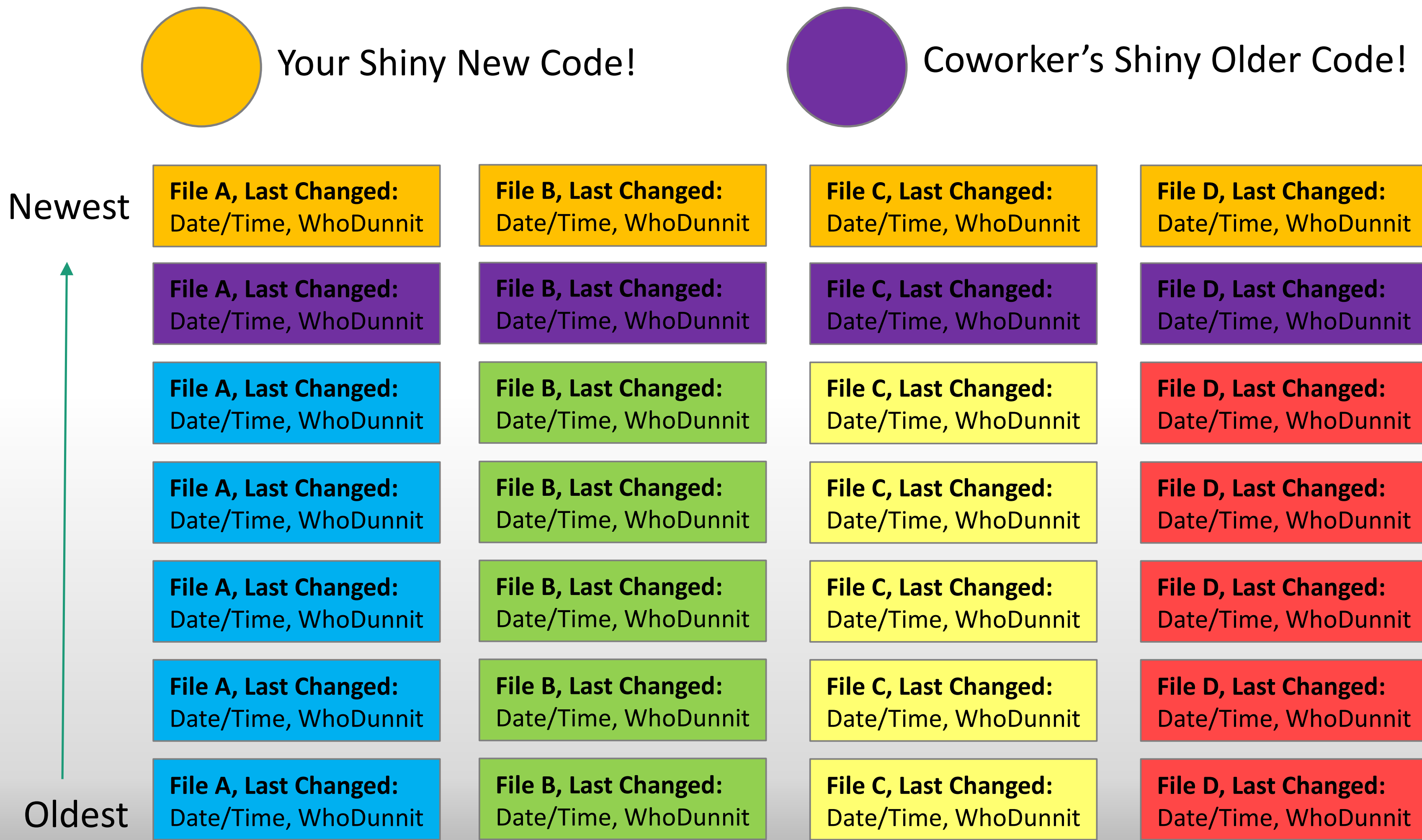
Mix-N-Matched Code

- Git might favor code based on write-time rather than merge-time
- Git might make ‘undesirable choices’ when two branches don’t have a common parent commit.

Trouble with Merge Conflicts

- Sometimes Git doesn’t think there’s a conflict (so Devs aren’t even prompted)

REBASE: Changes for 4 files in your Master branch



The following is inspired by real events.

MERGE

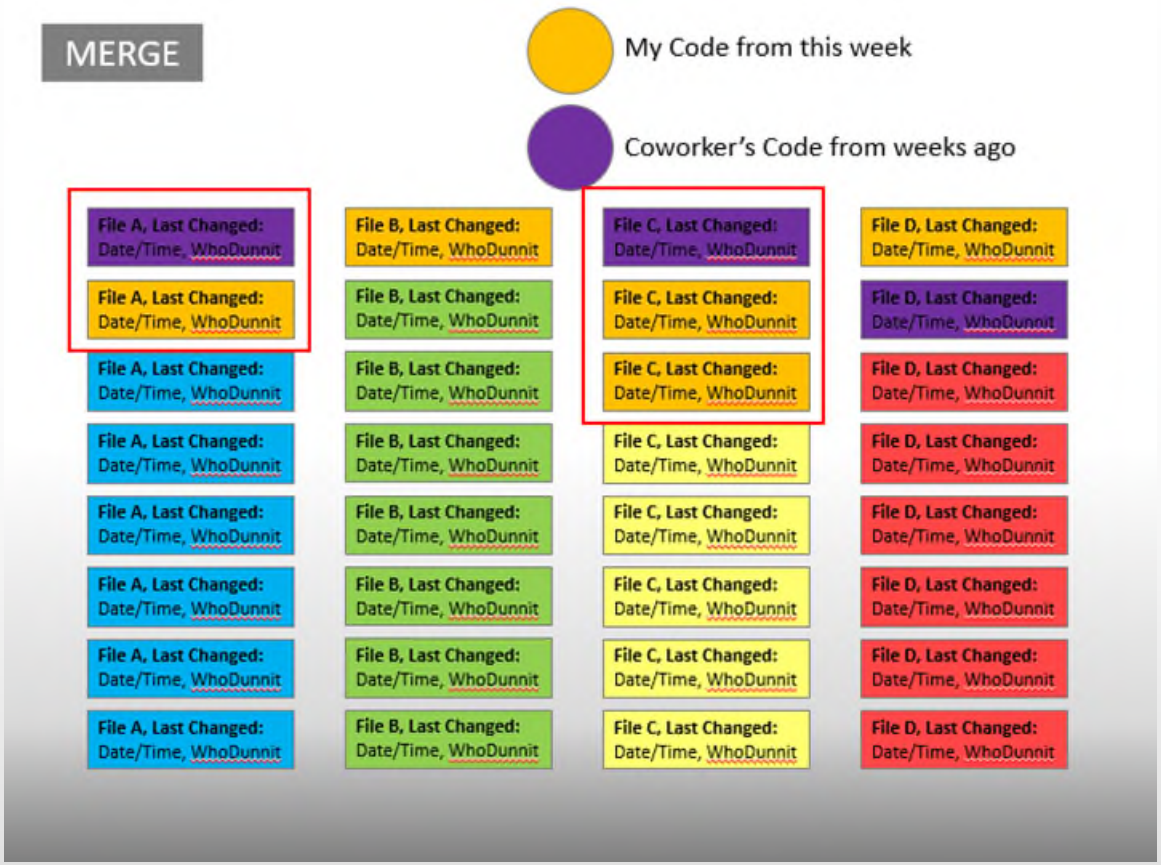
Changes for 4 files in your Master branch

My Shiny New Code!

Coworker's Shiny Older Code!

File A, Last Changed: Date/Time, WhoDunnit	**File B, Last Changed:** Date/Time, WhoDunnit	**File C, Last Changed:** Date/Time, WhoDunnit	**File D, Last Changed:** Date/Time, WhoDunnit
File A, Last Changed: Date/Time, WhoDunnit	**File B, Last Changed:** Date/Time, WhoDunnit	**File C, Last Changed:** Date/Time, WhoDunnit	**File D, Last Changed:** Date/Time, WhoDunnit
File A, Last Changed: Date/Time, WhoDunnit	**File B, Last Changed:** Date/Time, WhoDunnit	**File C, Last Changed:** Date/Time, WhoDunnit	**File D, Last Changed:** Date/Time, WhoDunnit
File A, Last Changed: Date/Time, WhoDunnit	**File B, Last Changed:** Date/Time, WhoDunnit	**File C, Last Changed:** Date/Time, WhoDunnit	**File D, Last Changed:** Date/Time, WhoDunnit
File A, Last Changed: Date/Time, WhoDunnit	**File B, Last Changed:** Date/Time, WhoDunnit	**File C, Last Changed:** Date/Time, WhoDunnit	**File D, Last Changed:** Date/Time, WhoDunnit
File A, Last Changed: Date/Time, WhoDunnit	**File B, Last Changed:** Date/Time, WhoDunnit	**File C, Last Changed:** Date/Time, WhoDunnit	**File D, Last Changed:** Date/Time, WhoDunnit
File A, Last Changed: Date/Time, WhoDunnit	**File B, Last Changed:** Date/Time, WhoDunnit	**File C, Last Changed:** Date/Time, WhoDunnit	**File D, Last Changed:** Date/Time, WhoDunnit

So what happened?



Pull happened.

`git pull` == git merge

"In its default mode, git pull is shorthand for 'git fetch' followed by 'git merge FETCH_HEAD'."

- Git Documentation

Fetch, Pull
Rebase, Merge

…And also team communication…

I hope you know a bit more
than you did before