Git:
Fetch, Pull
Rebase, Merge

Lauren Billington

Legislative Information Services

Colorado State Legislature

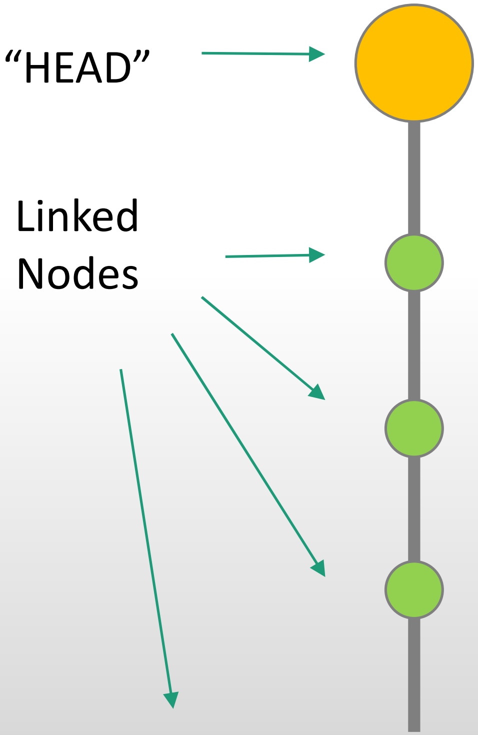
My Team's recent Gitastrophy

a PERFECT example!

Learned to code:
Use ‘git merge’

First dev job:
Use rebase workflow

The Classic Visualization



“HEAD”

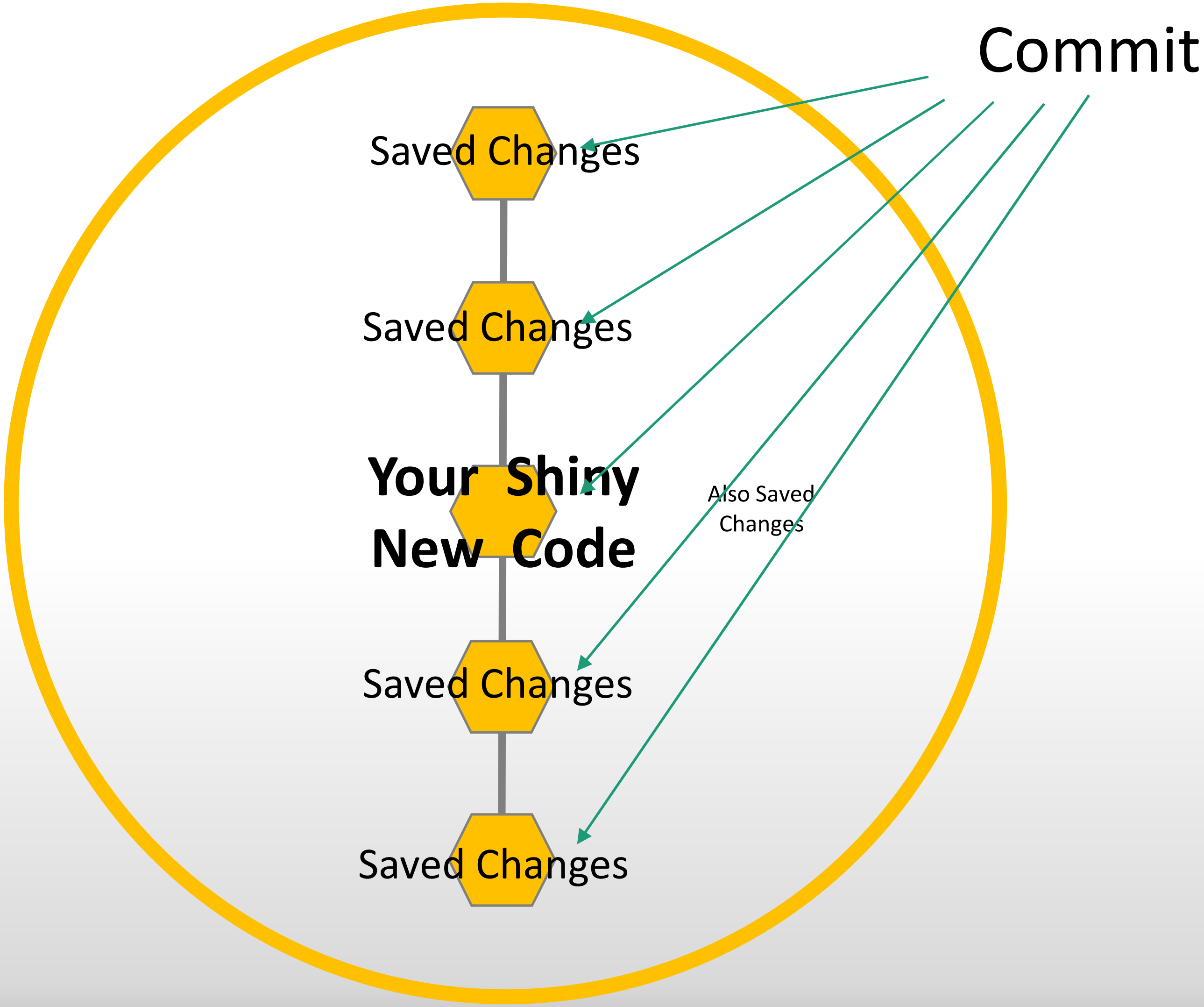
Linked Nodes

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

Your Shiny New Code



Commit

Saved Changes

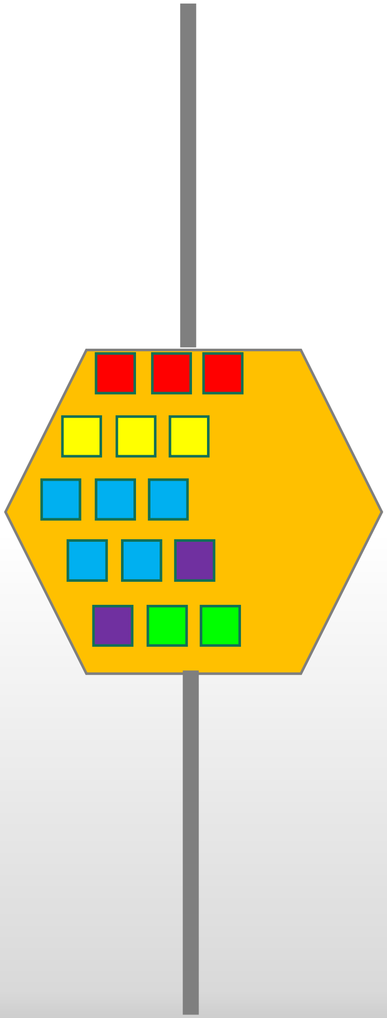
Saved Changes

Your Shiny New Code

Also Saved Changes

Saved Changes

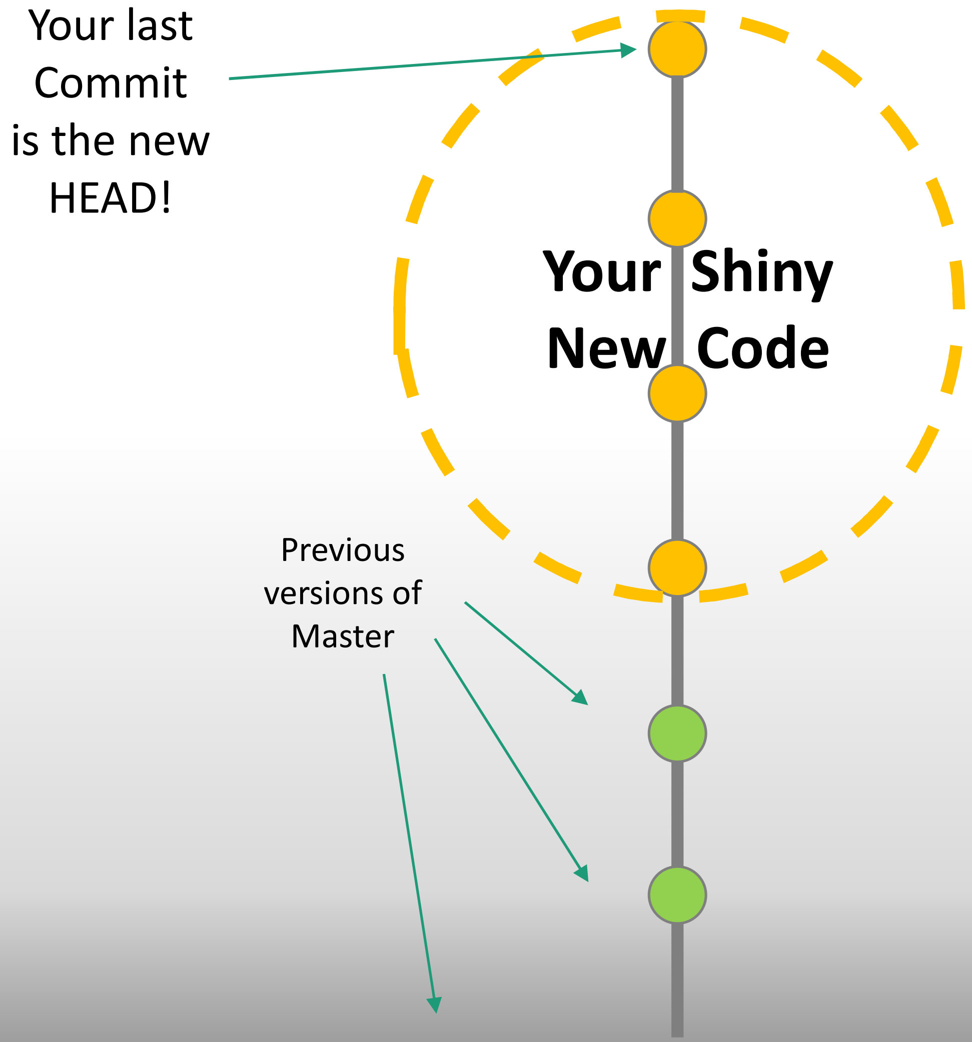
Saved Changes



1 Commit **=**

1 copy of the **whole codebase**
including your new changes

The simplest use case



Your last Commit is the new HEAD!

Your Shiny New Code

Previous versions of Master

Rebasing vs Merge

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

Rebasing vs Merge

- ‘git merge’ (no args)
  - ‘hey Git make decisions for me plz’
  - Attempts one of 6 merging strategies:
    - Fast Forward
    - Recursive
    - Ours
    - Octopus
    - Resolve
    - Subtree

Flash Forward: Gitastrophy

- Sole developer
- Using rebase workflow
- Lots of refactors and new features

Flash Forward: Gitastrophy

- Test suite passed
- Code passed QA

Flash Forward: Gitastrophy

- Test suite passed
- Code passes QA
- Prod. Blows. Up.

What Happened?

- Server logs
- Commits and git history
- Terminal command history

Repo Master != Prod Master

Culprit?

‘git pull’

Culprit?

~~‘git pull’~~

COMMUNICATION

Culprit (technically)?

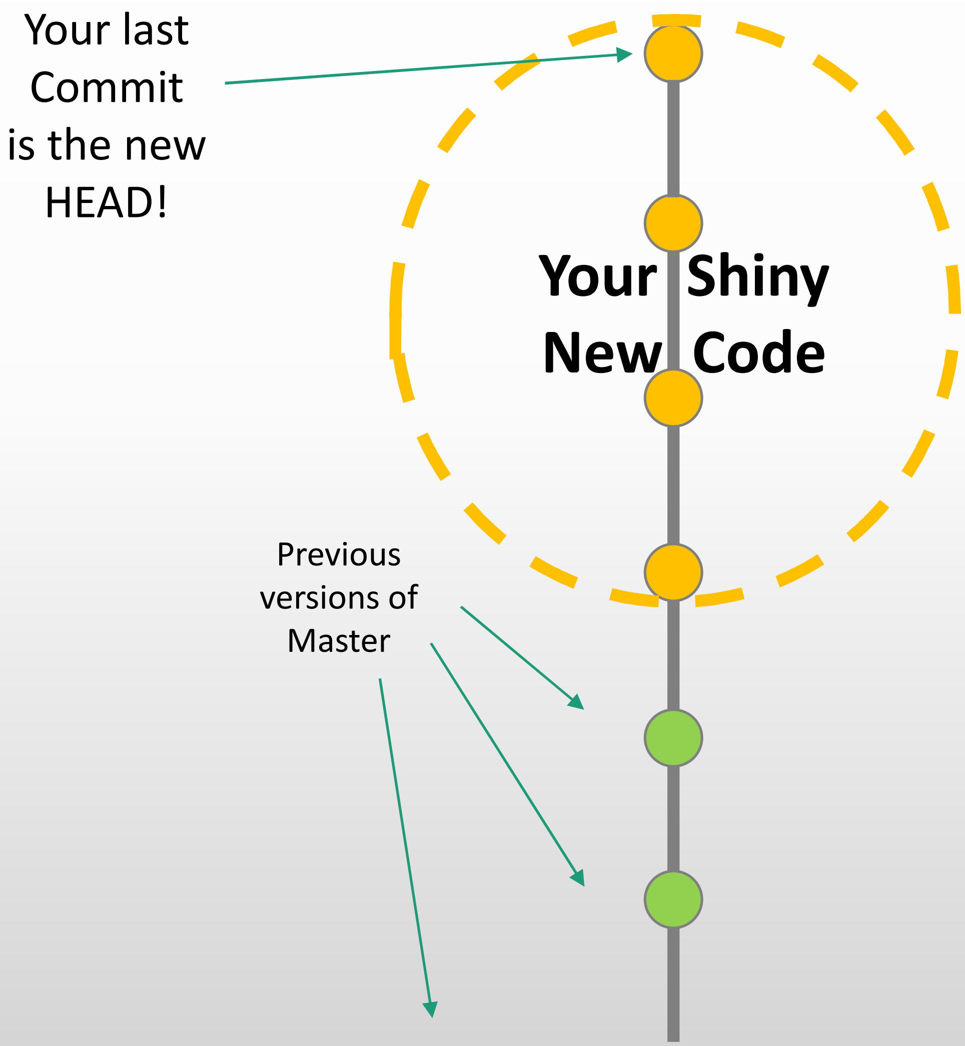
‘git pull’

`git pull` == git merge

“In its default mode, git pull is shorthand for
‘git fetch’ followed by ‘git merge FETCH_HEAD’.”

- Git Documentation

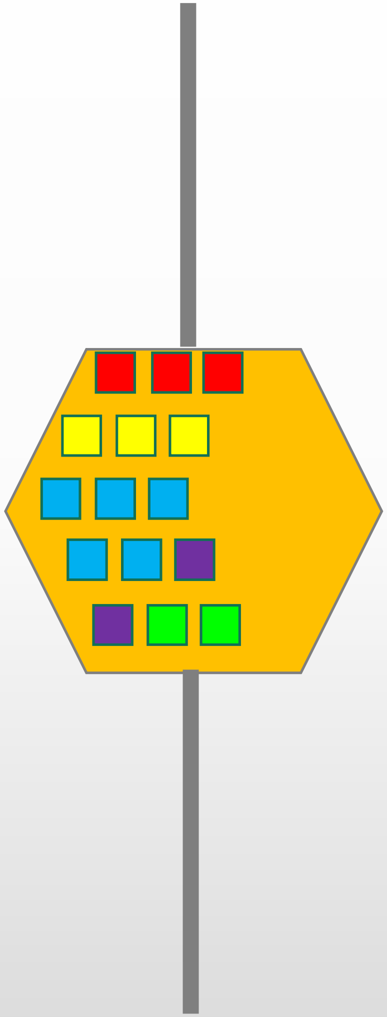
Using ‘git rebase -i’



Your last Commit is the new HEAD!

Your Shiny New Code

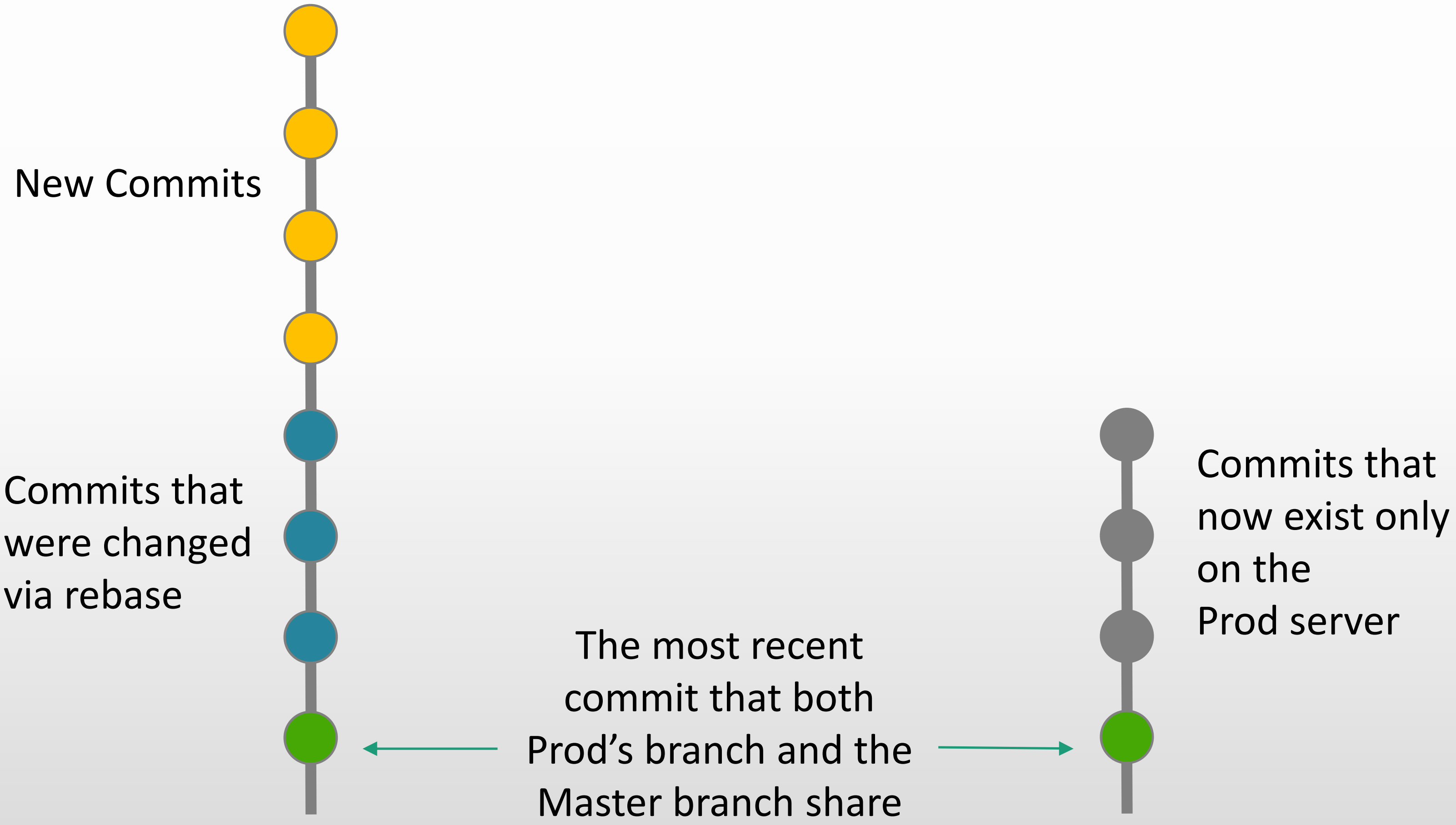
Previous versions of Master



1 Commit **=**

1 copy of the **whole codebase** *including* your new changes

Repo vs Prod



New Commits

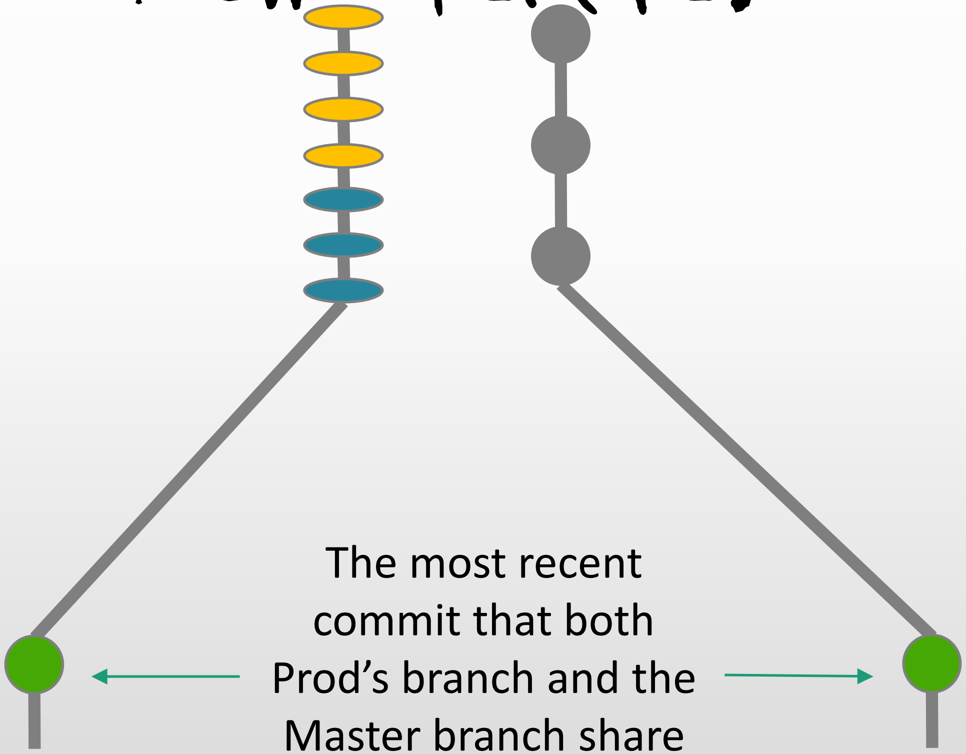
Commits that were changed via rebase

The most recent commit that both Prod's branch and the Master branch share

Commits that now exist only on the Prod server

Repo vs Prod

Now MERGE!



The most recent commit that both Prod's branch and the Master branch share

The Fix

```
‘git fetch upstream master &&
git reset --hard upstream/master’
```

Kinda anticlimactic ending…

Fetch, Pull
Rebase, Merge

…And also team communication…

I hope you know a bit more
than you did before

Thank you!