04

Gitshop

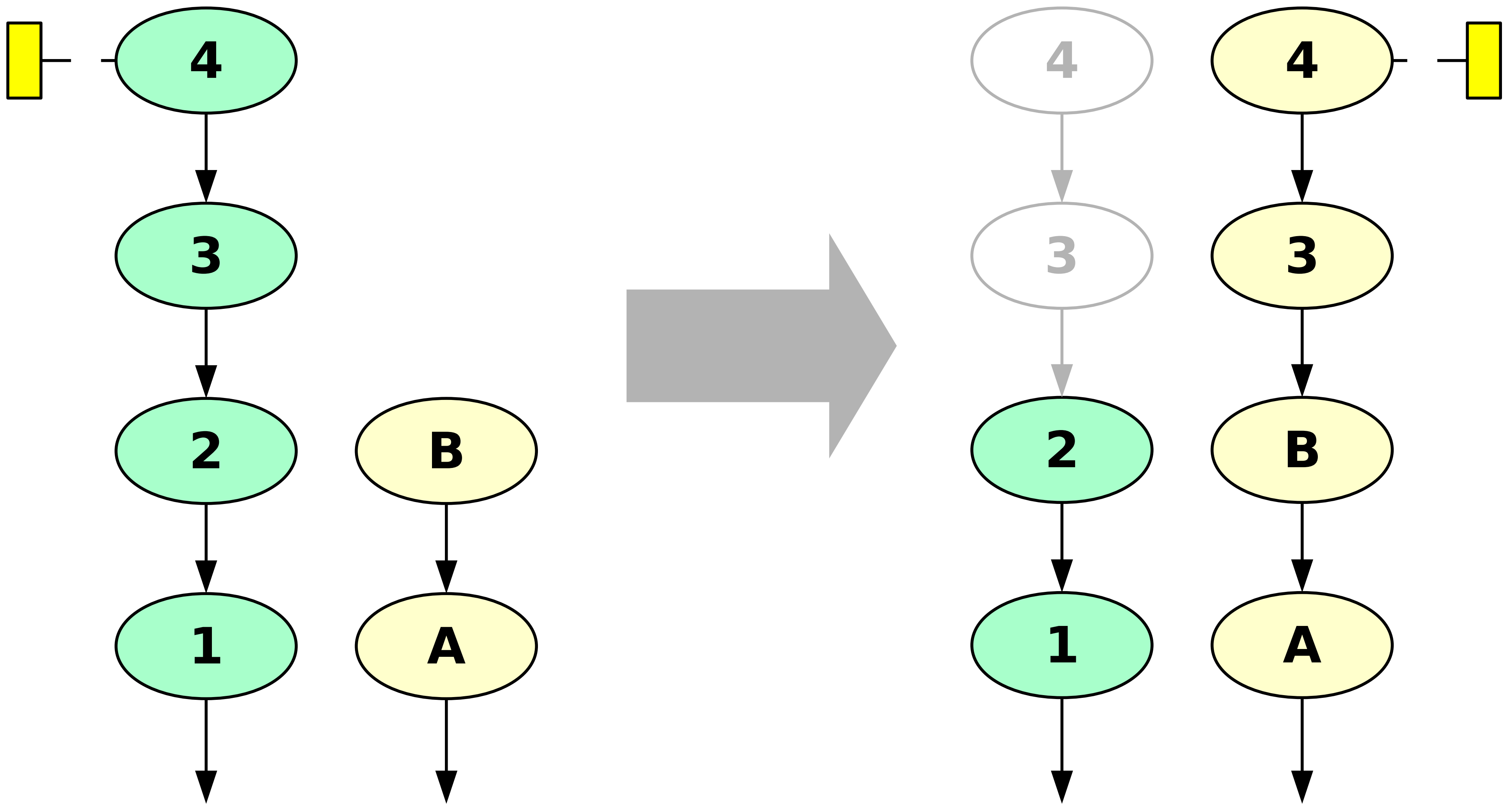
Git - rebase

Jiří Jabůrek (jjaburek)
Red Hat 2012

Rebase = ?

Rebase = ?

- Change of "base" of a group of commits

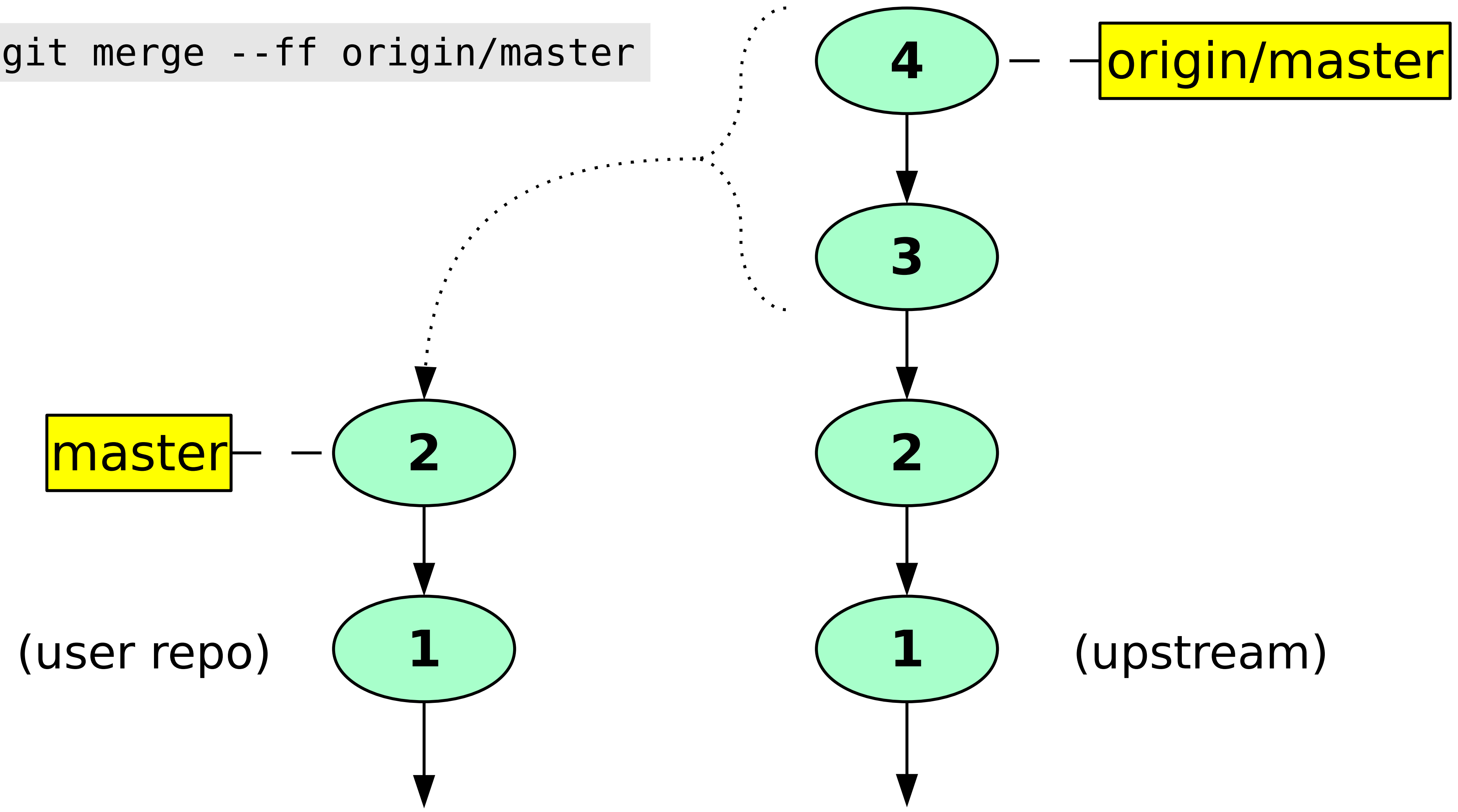


Fast-forward merge

Fast-forward merge

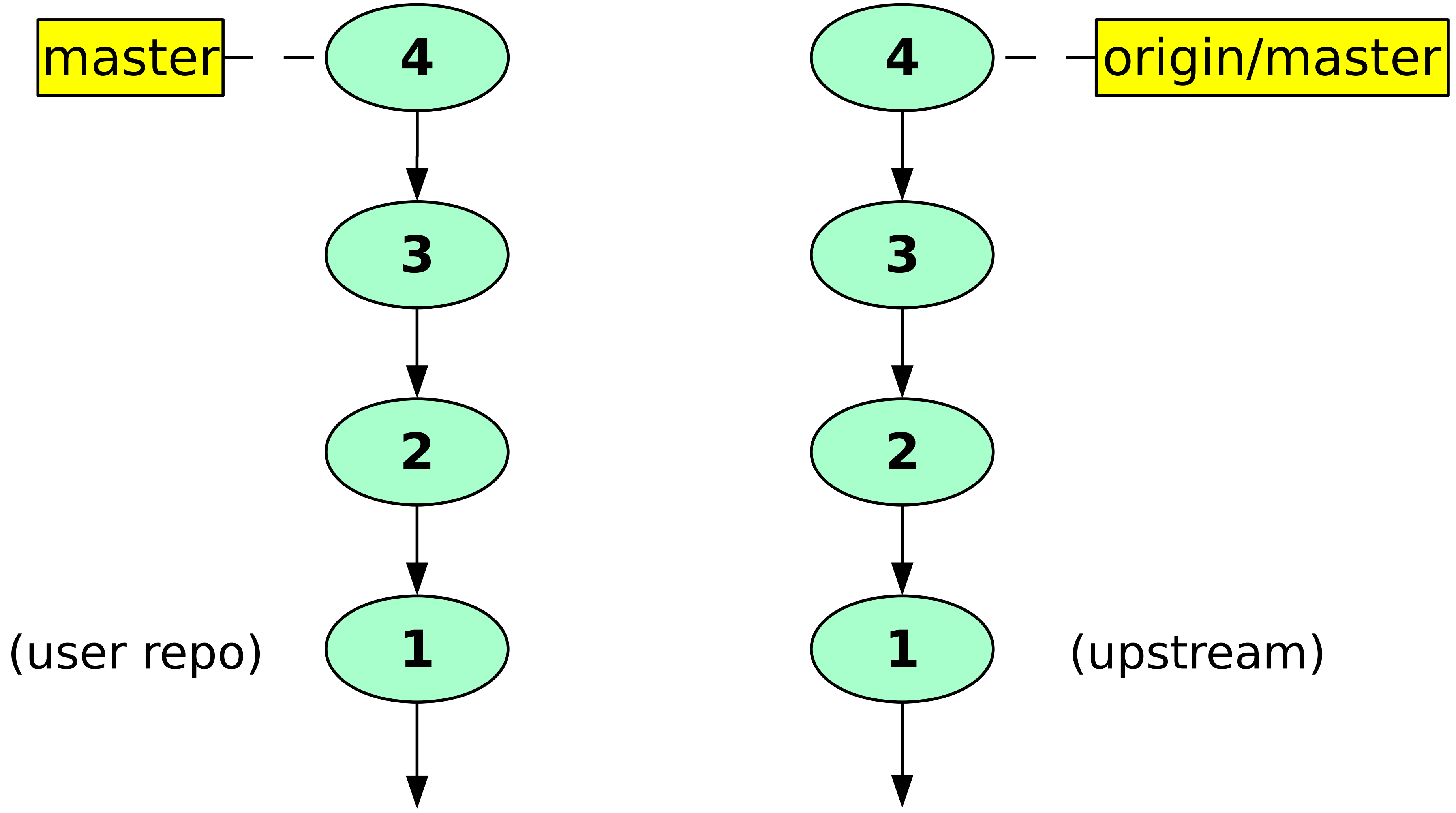
- "Get new code from upstream"

```
git merge --ff origin/master
```



Fast-forward merge

- "Get new code from upstream"

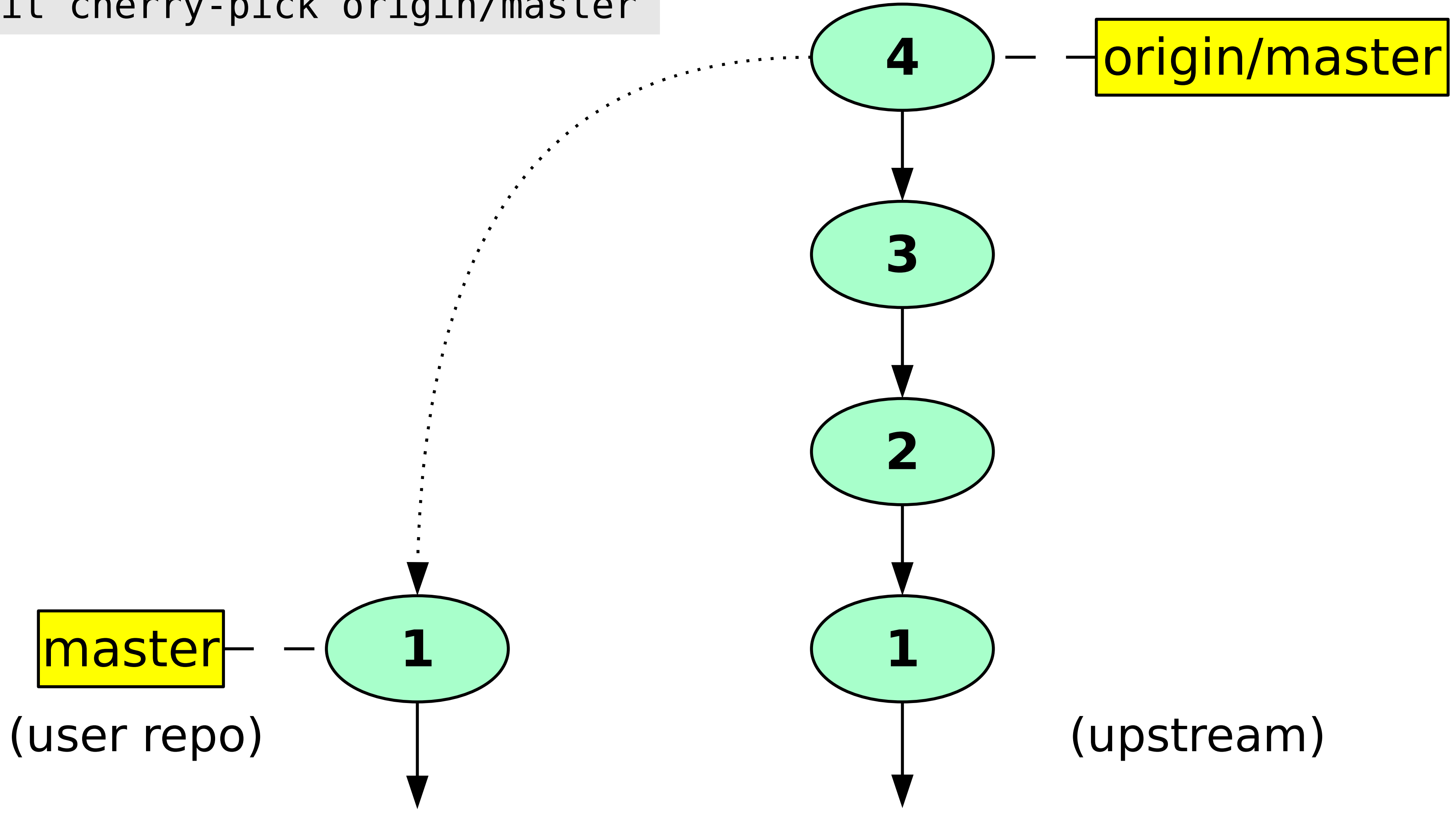


Cherry picking

Cherry picking

- Used to "copy" selected commits

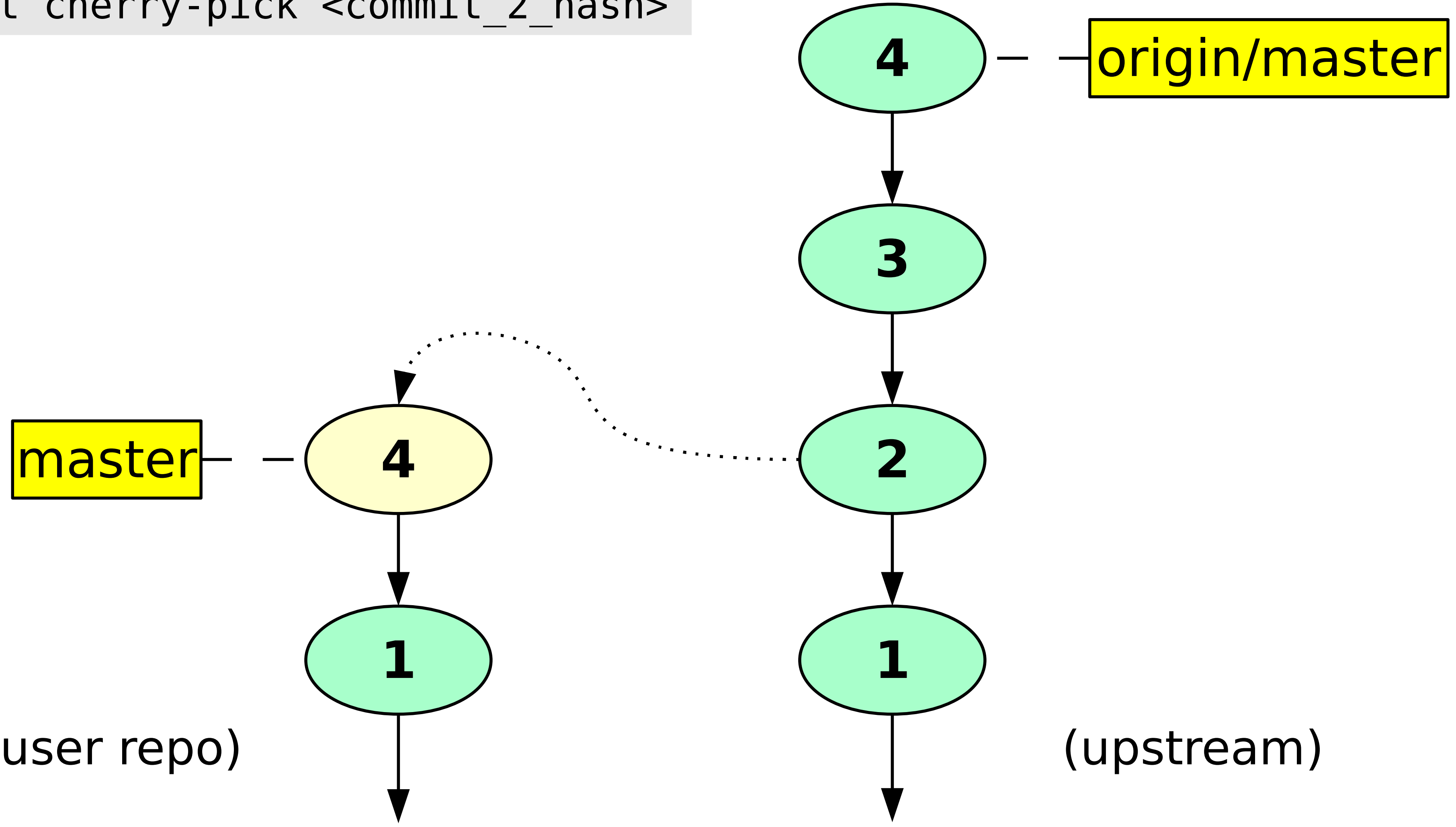
```
git cherry-pick origin/master
```



Cherry picking

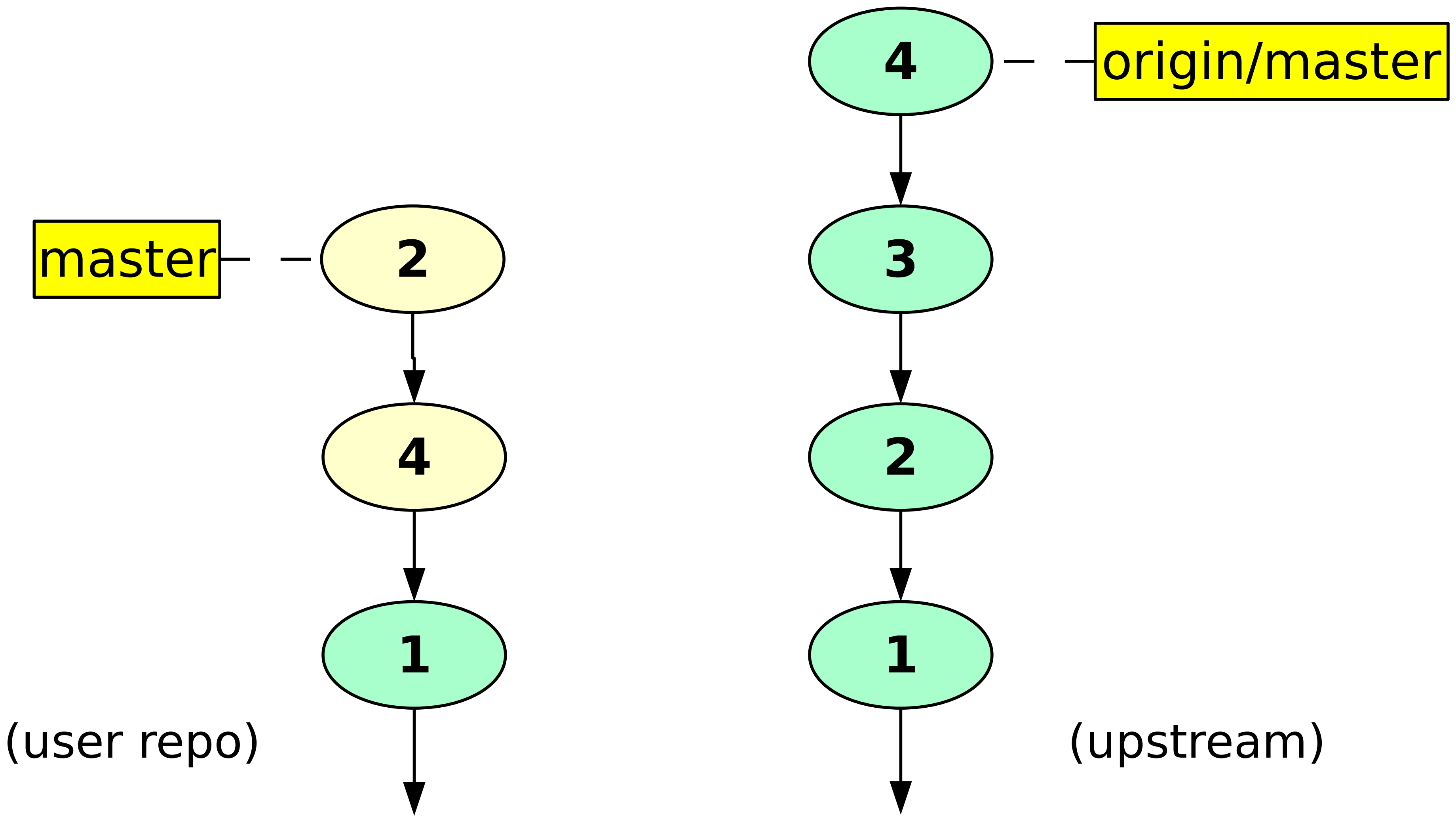
- Used to "copy" selected commits

```
git cherry-pick <commit_2_hash>
```



Cherry picking

- Used to "copy" selected commits

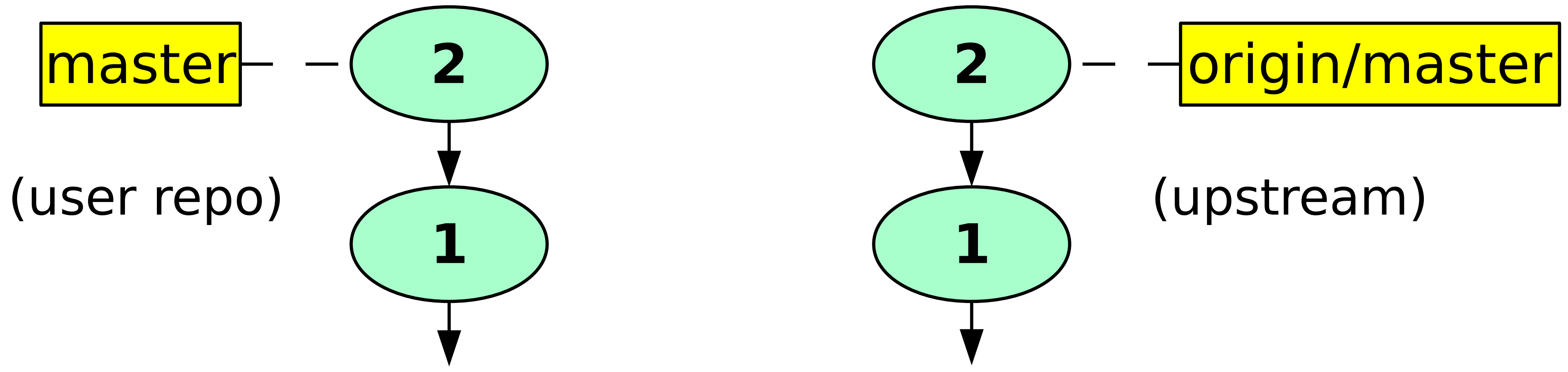


git-rebase

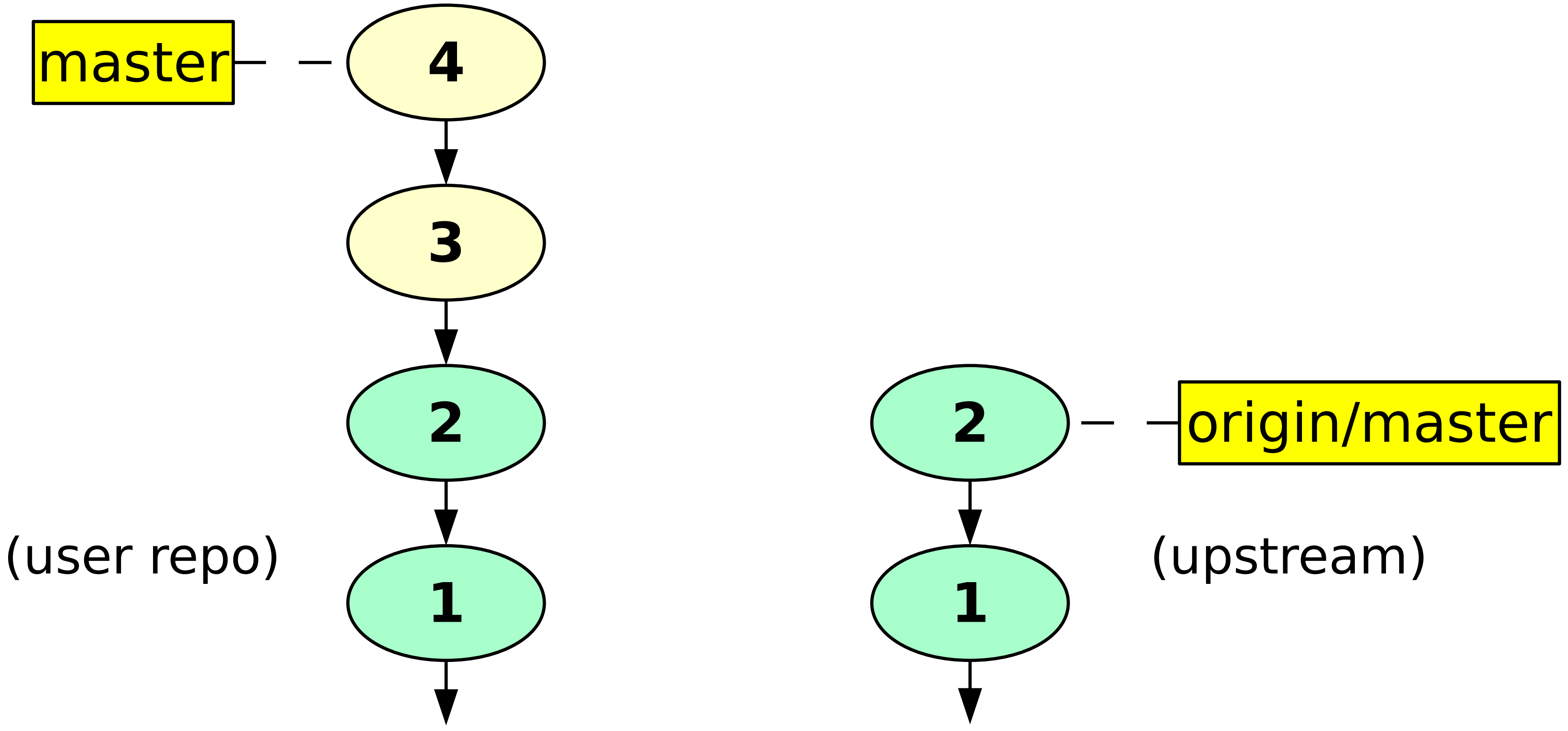
git-rebase

- use cases
  - git rebase [upstream]
    - rebase current branch to "upstream" branch
  - git rebase --onto [new] [start]
    - take all commits from "start" up to the current branch, rebase them on top of "new" commit/branch
  - git rebase -i [refspec]
    - does an interactive rebase of the current branch against "refspec", which should be in the history of the current branch, like "HEAD~5"
  - git rebase --continue | --abort | --skip
    - useful when rebase is interrupted (ie. conflicts)

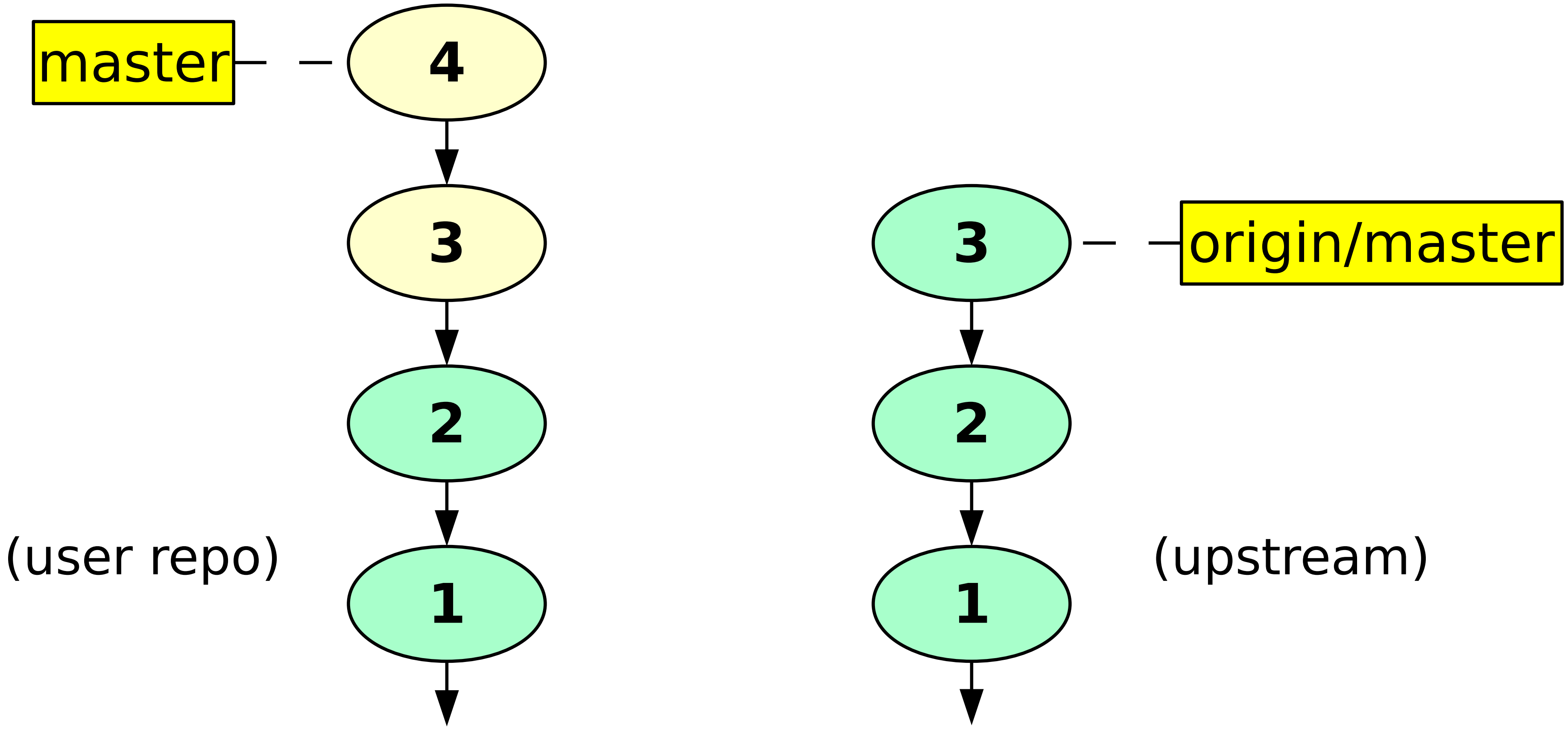
git-rebase [upstream]



git-rebase [upstream]

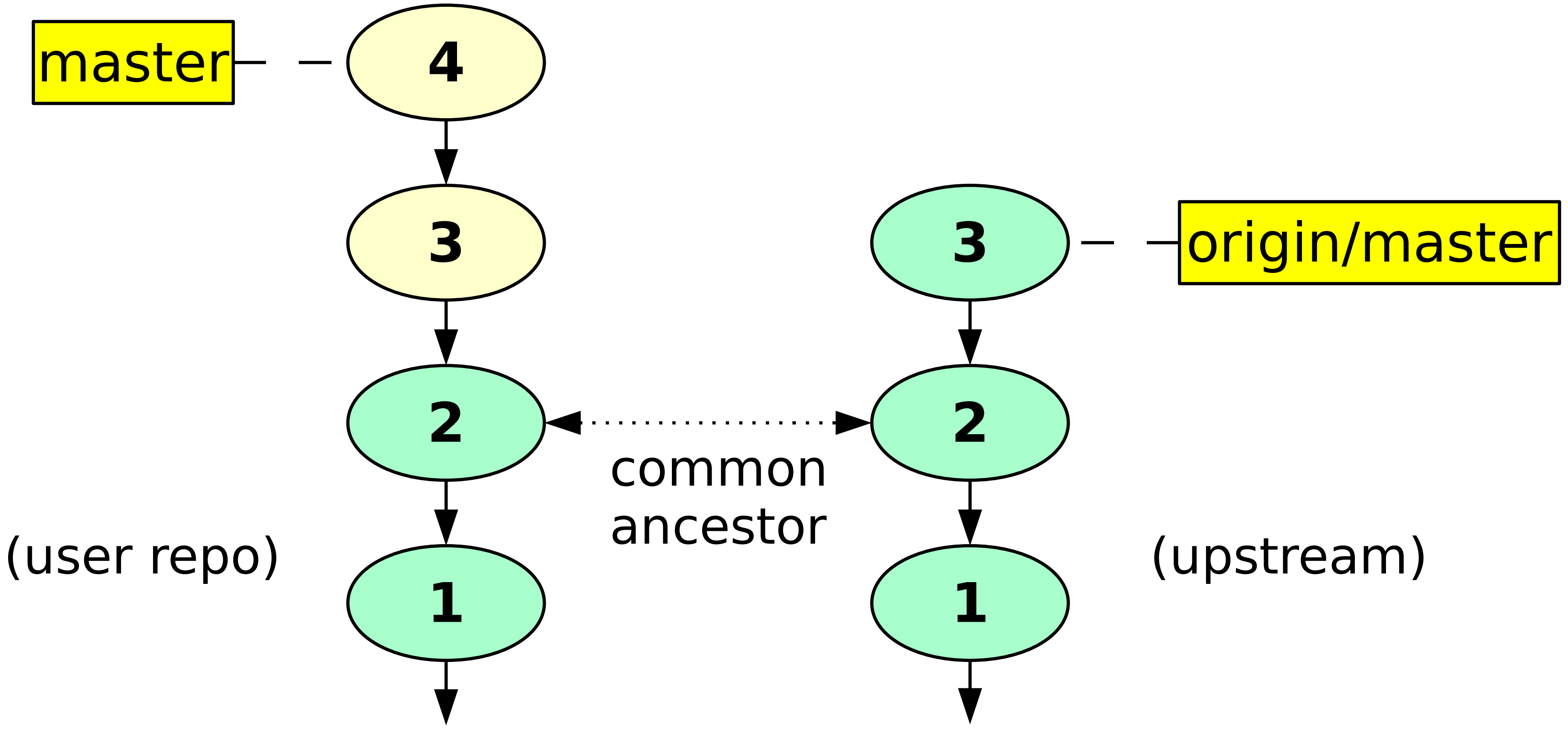


git-rebase [upstream]



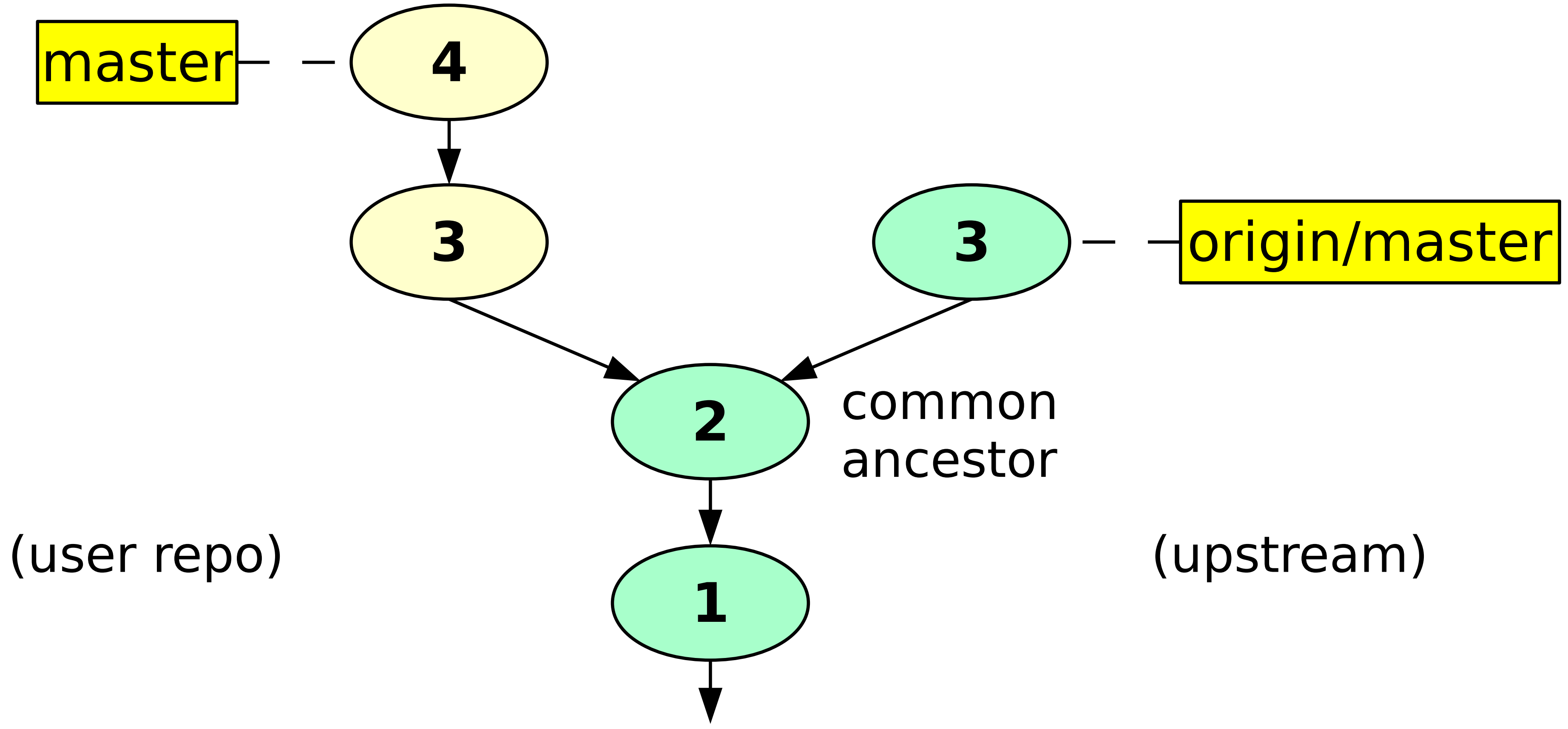
git-rebase [upstream]

```
git rebase origin/master
```



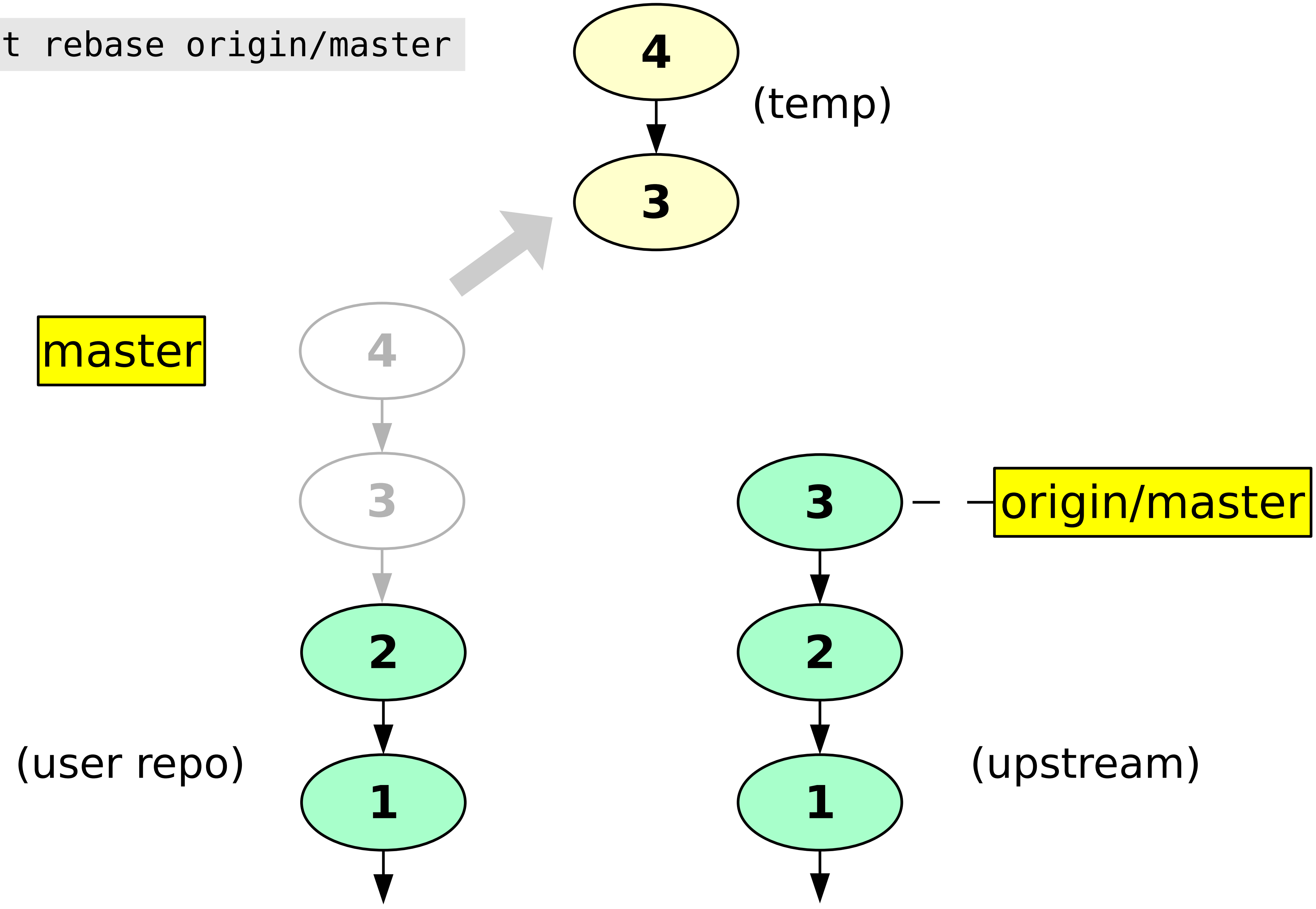
git-rebase [upstream]

```
git rebase origin/master
```



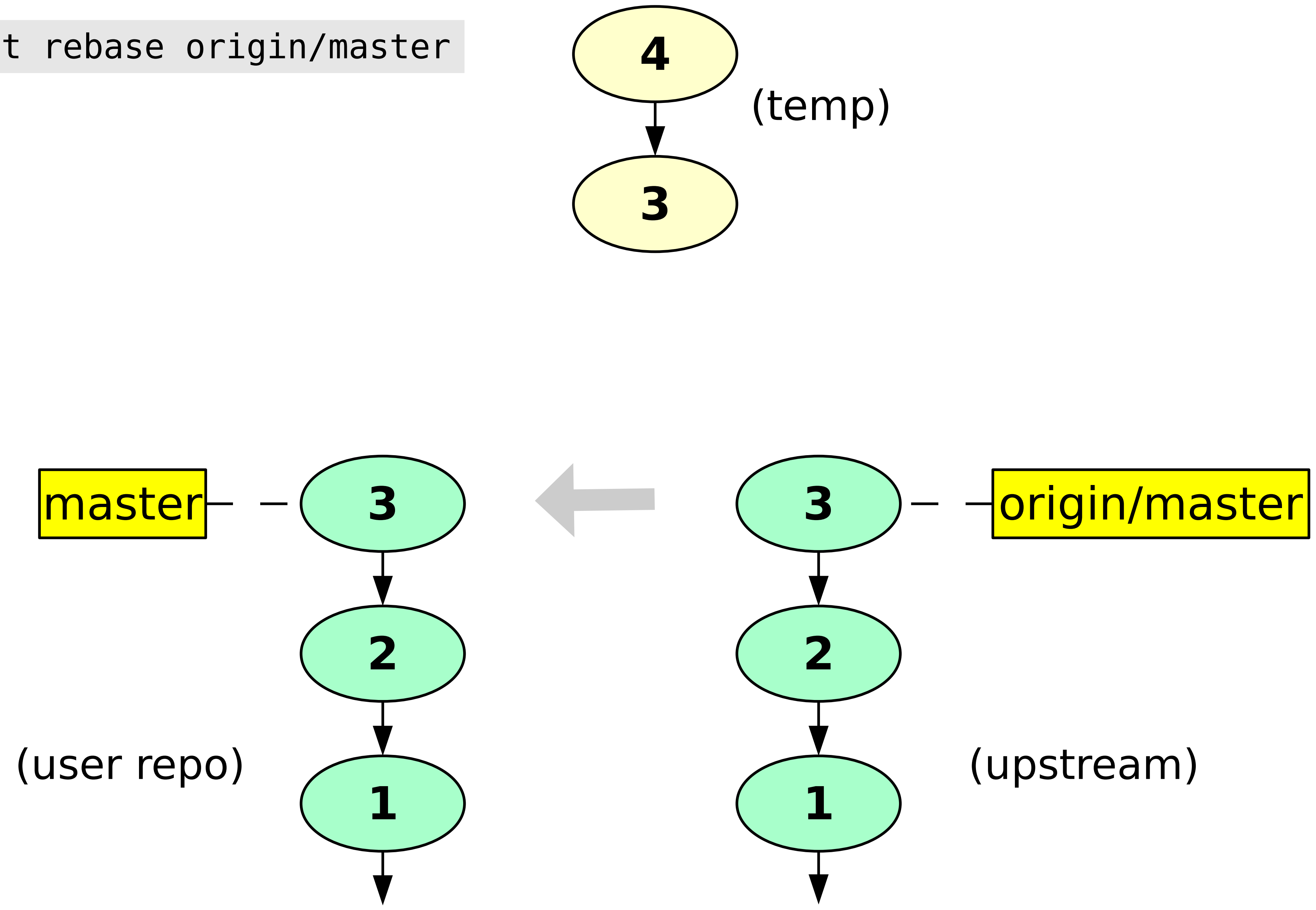
git-rebase [upstream]

```
git rebase origin/master
```



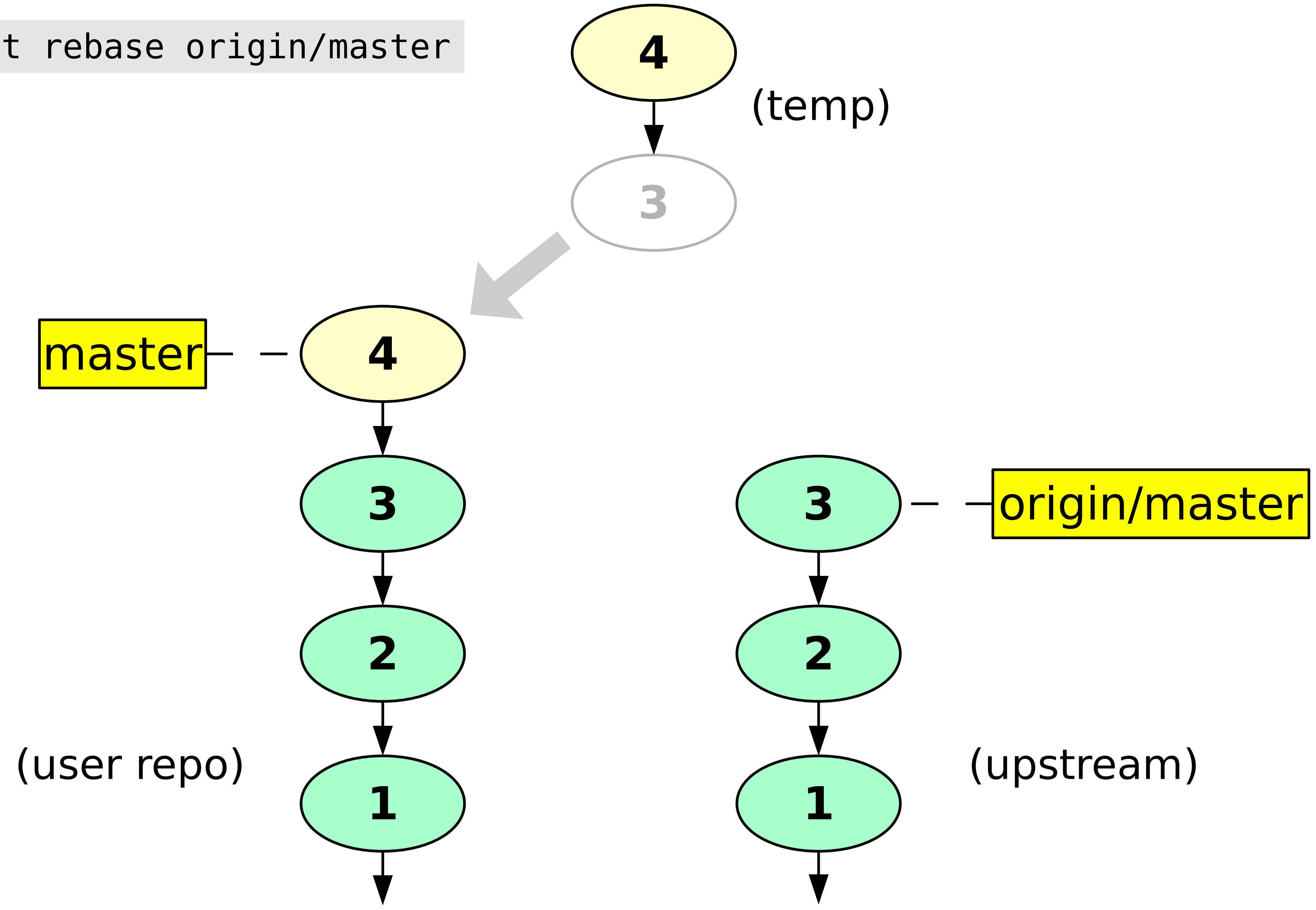
git-rebase [upstream]

```
git rebase origin/master
```



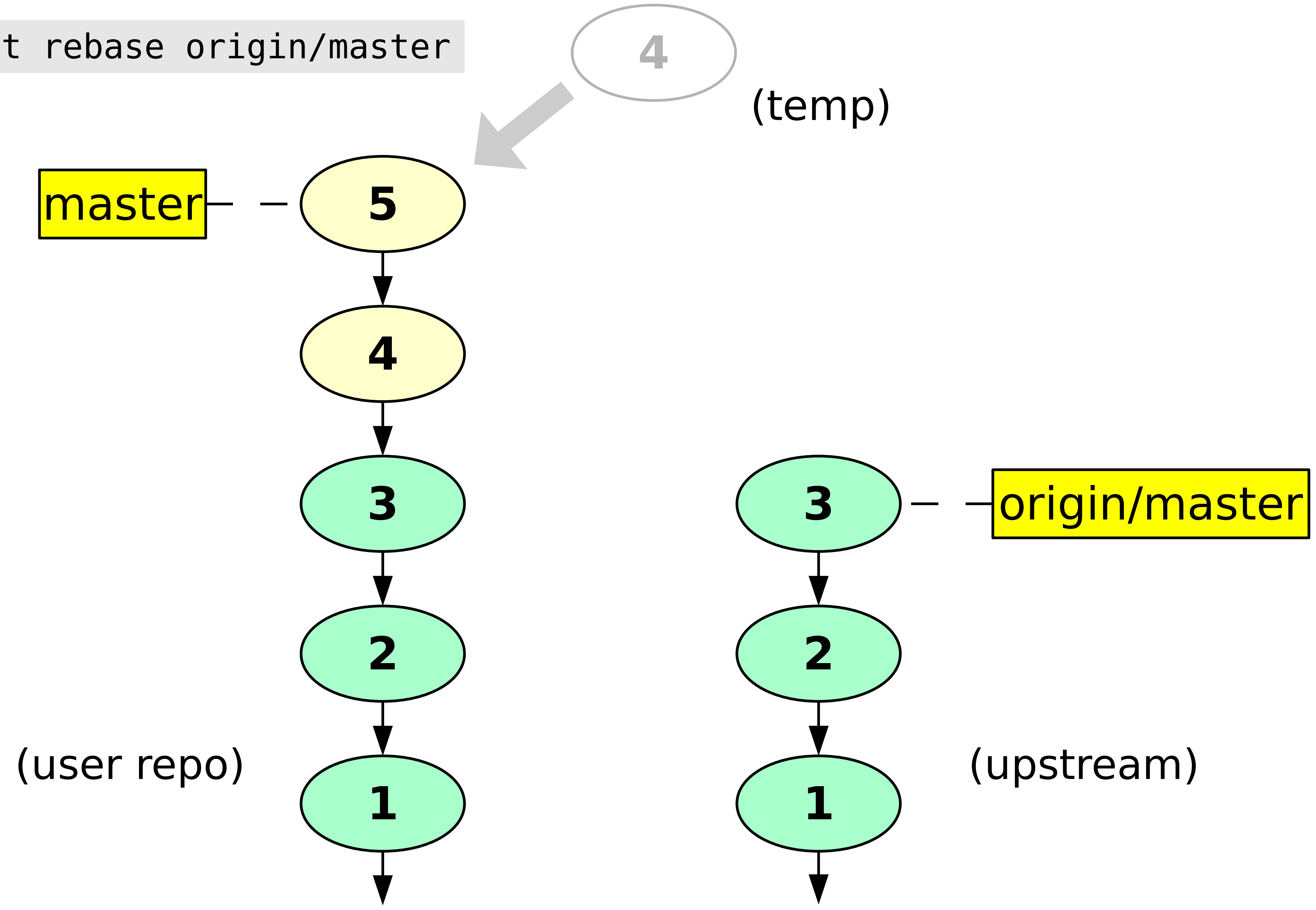
git-rebase [upstream]

```
git rebase origin/master
```

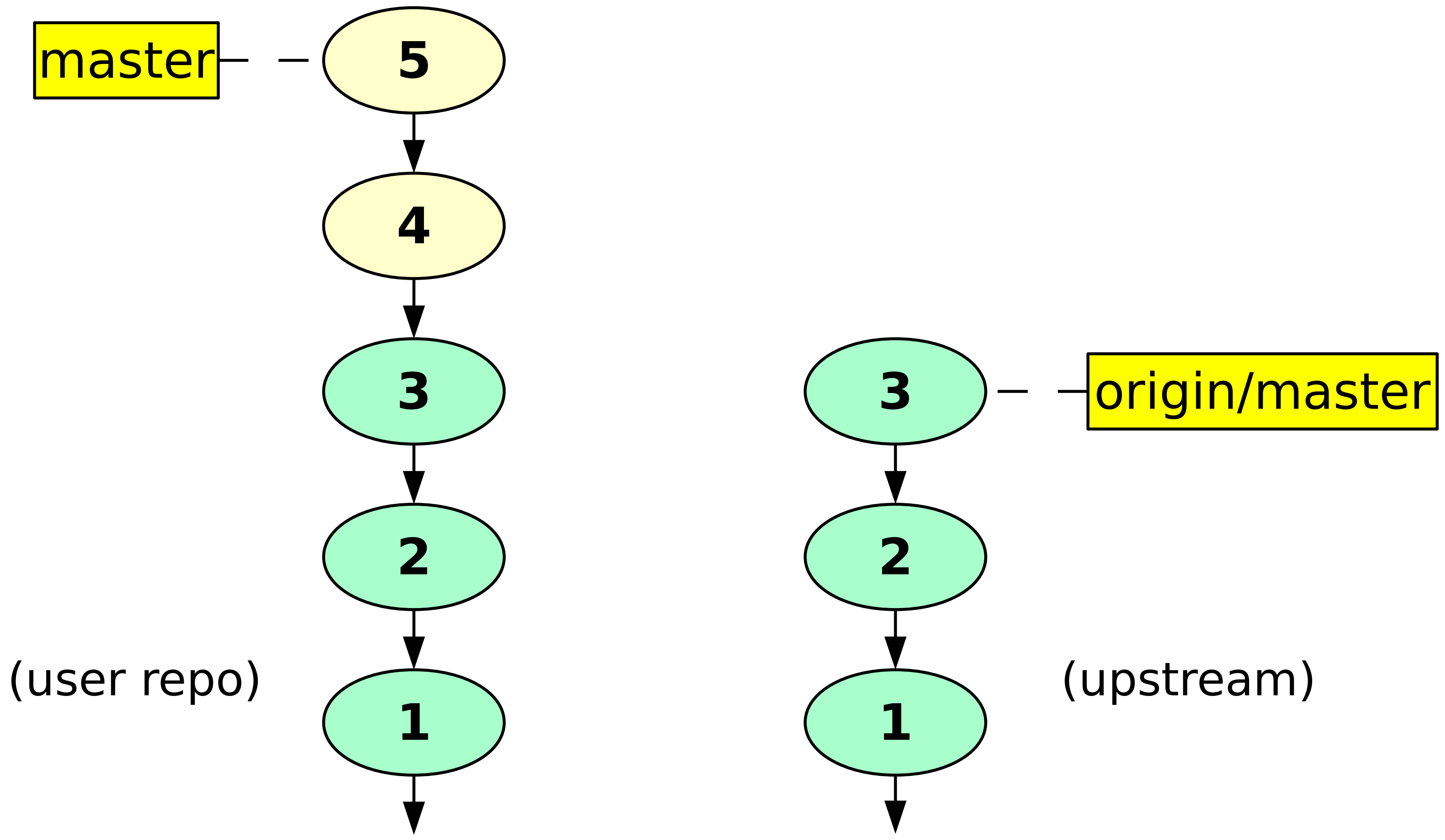


git-rebase [upstream]

```
git rebase origin/master
```



git-rebase [upstream]

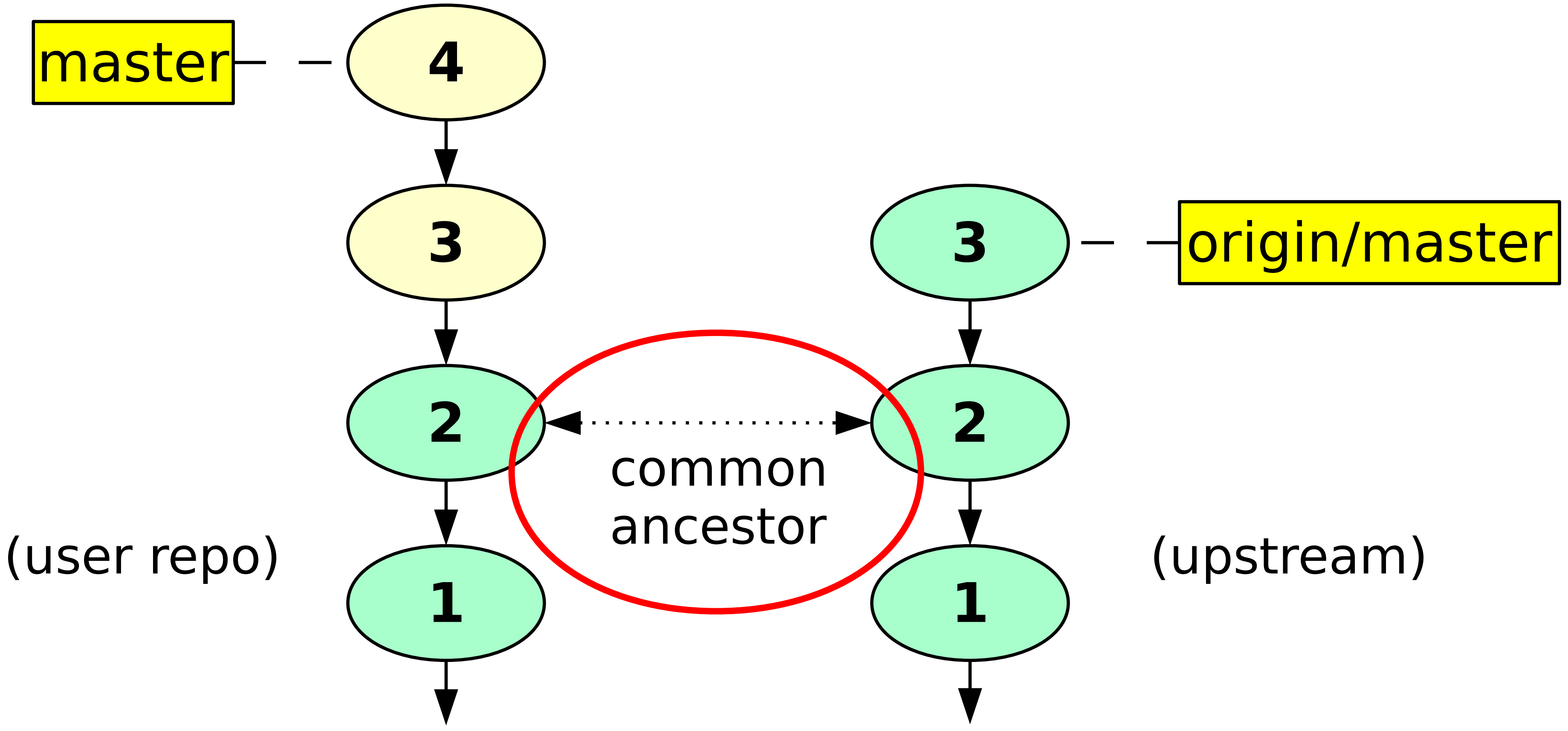


git-rebase

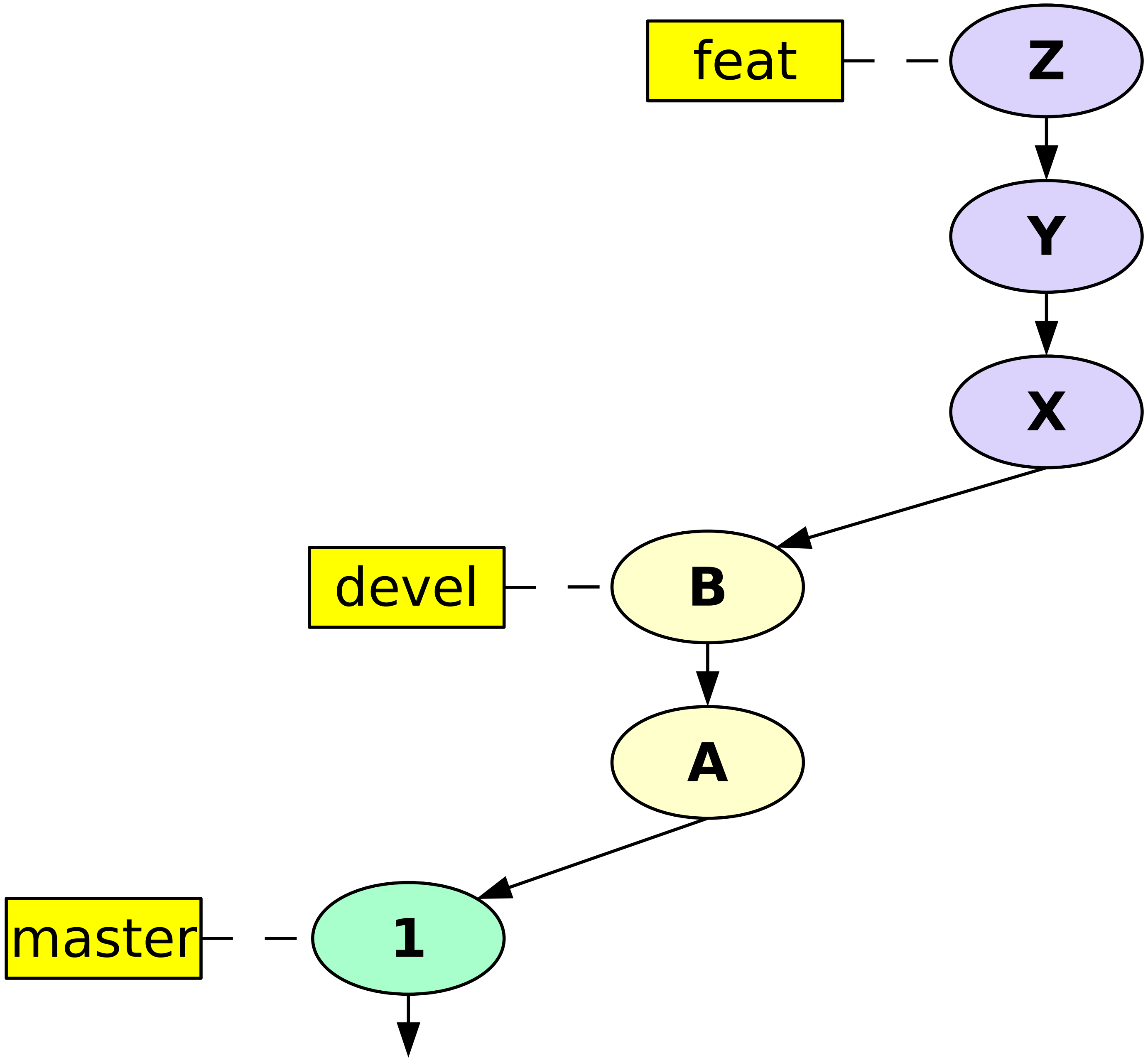
- use cases
  - git rebase [upstream]
    - rebase current branch to "upstream" branch
  - git rebase --onto <new> [start]
    - take all commits from "start" up to the current branch, rebase them on top of "new" commit/branch
  - git rebase -i <refspec>
    - does an interactive rebase of the current branch against "refspec", which should be in the history of the current branch, like "HEAD~5"
  - git rebase --continue | --abort | --skip
    - useful when rebase is interrupted (ie. conflicts)

git-rebase --onto <new> [start]

- manual specification of starting point
  - (instead of common ancestor)

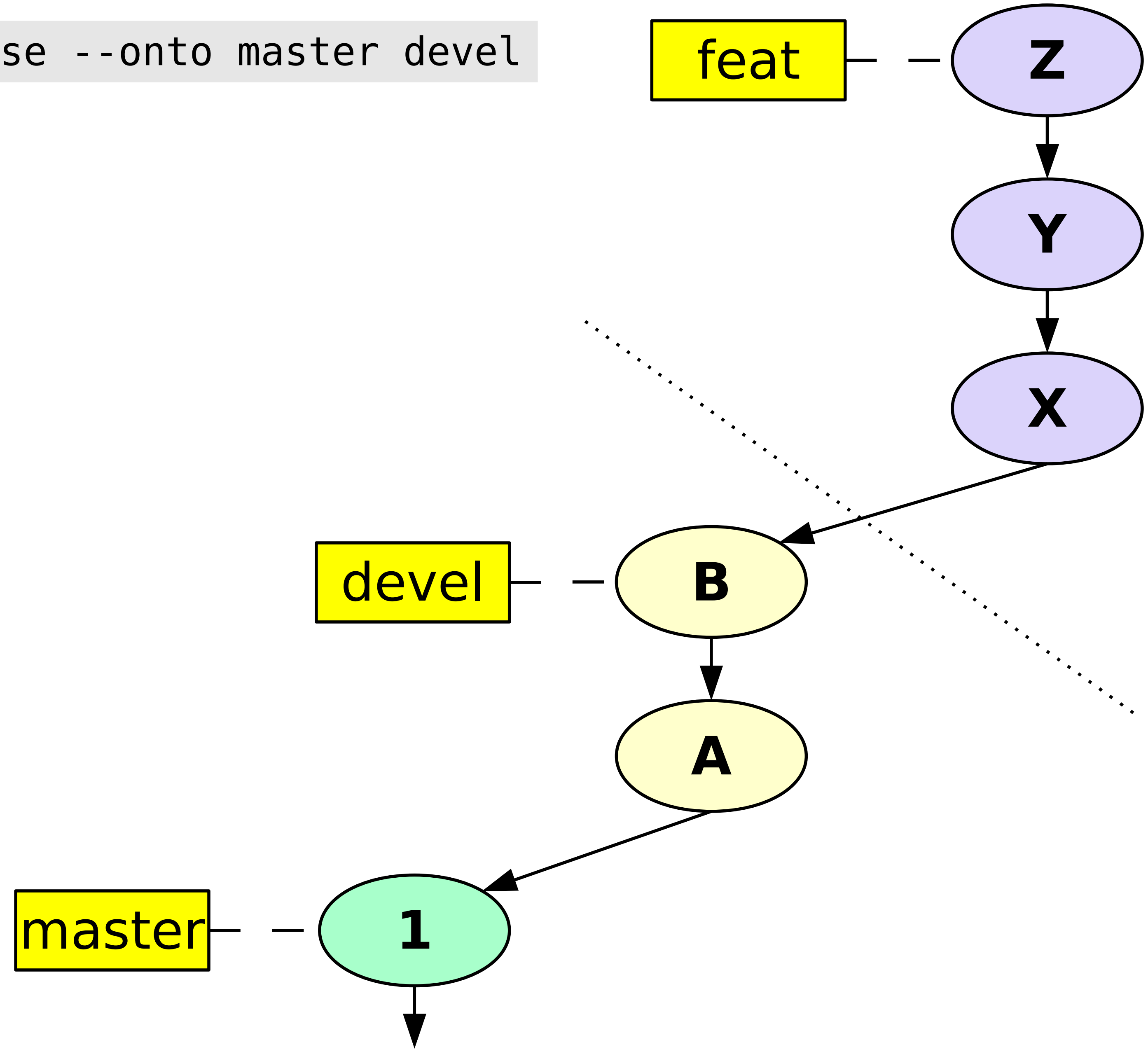


git-rebase --onto <new> [start]

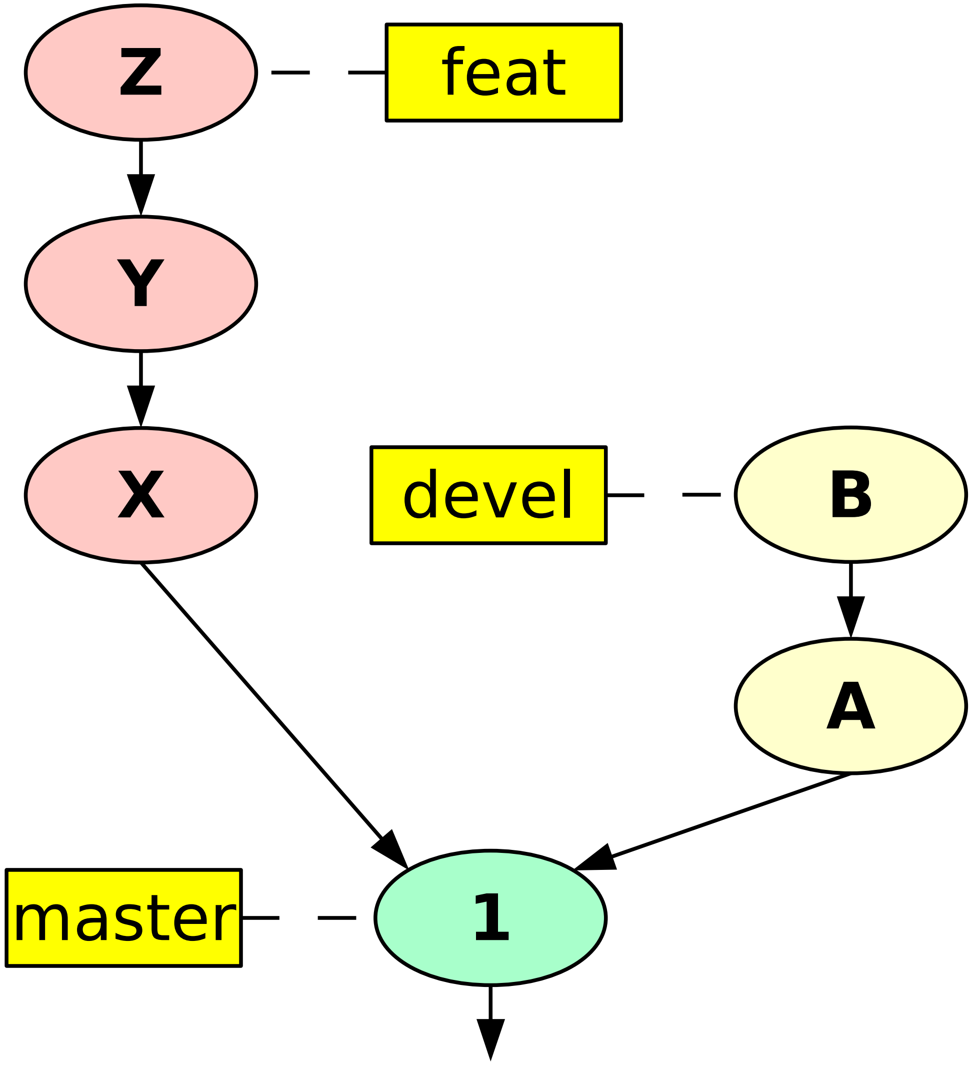


git-rebase --onto <new> [start]

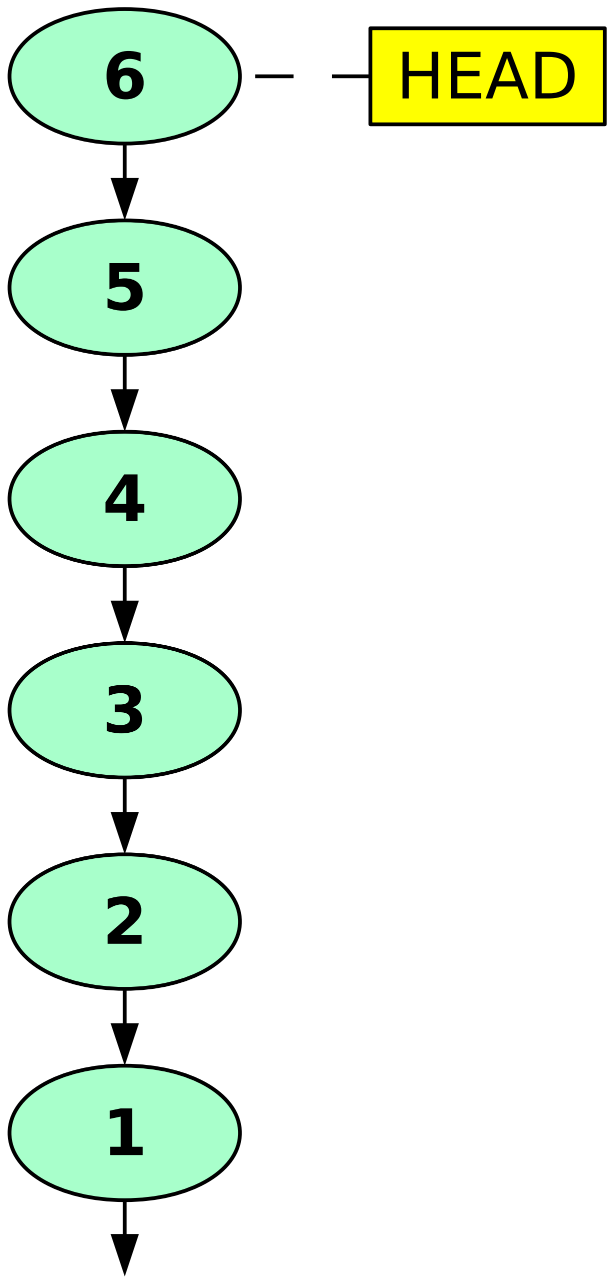
```
git rebase --onto master devel
```



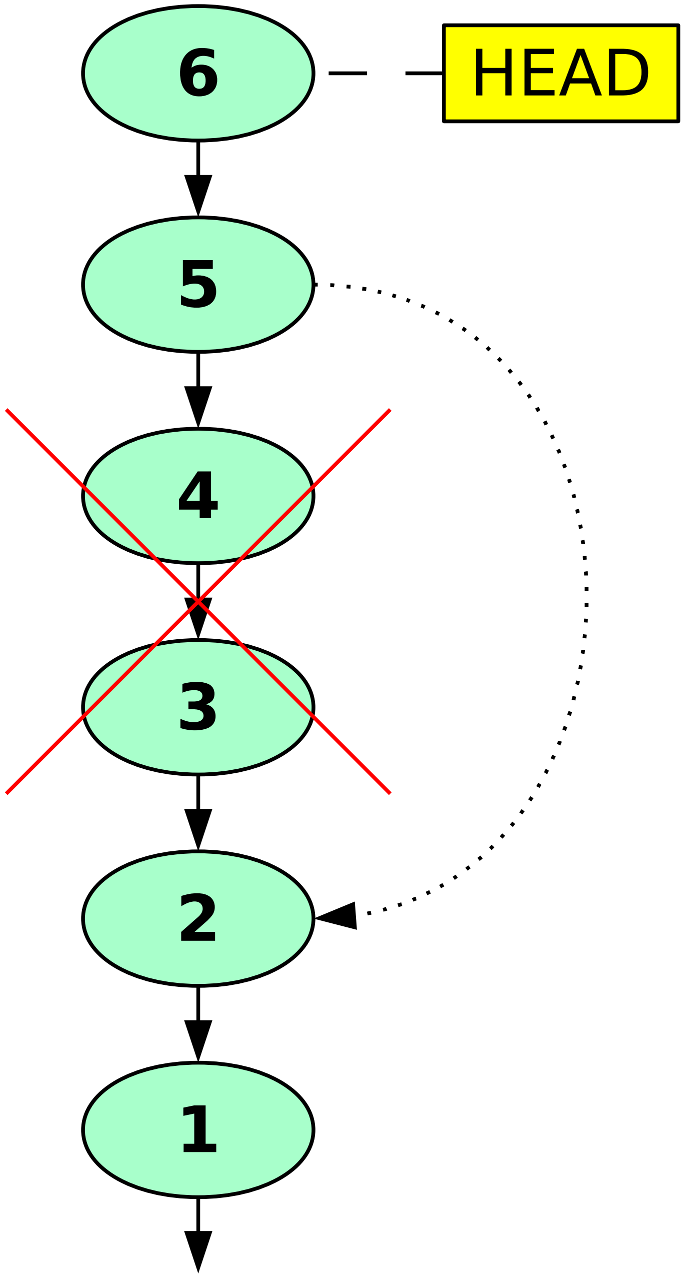
git-rebase --onto <new> [start]



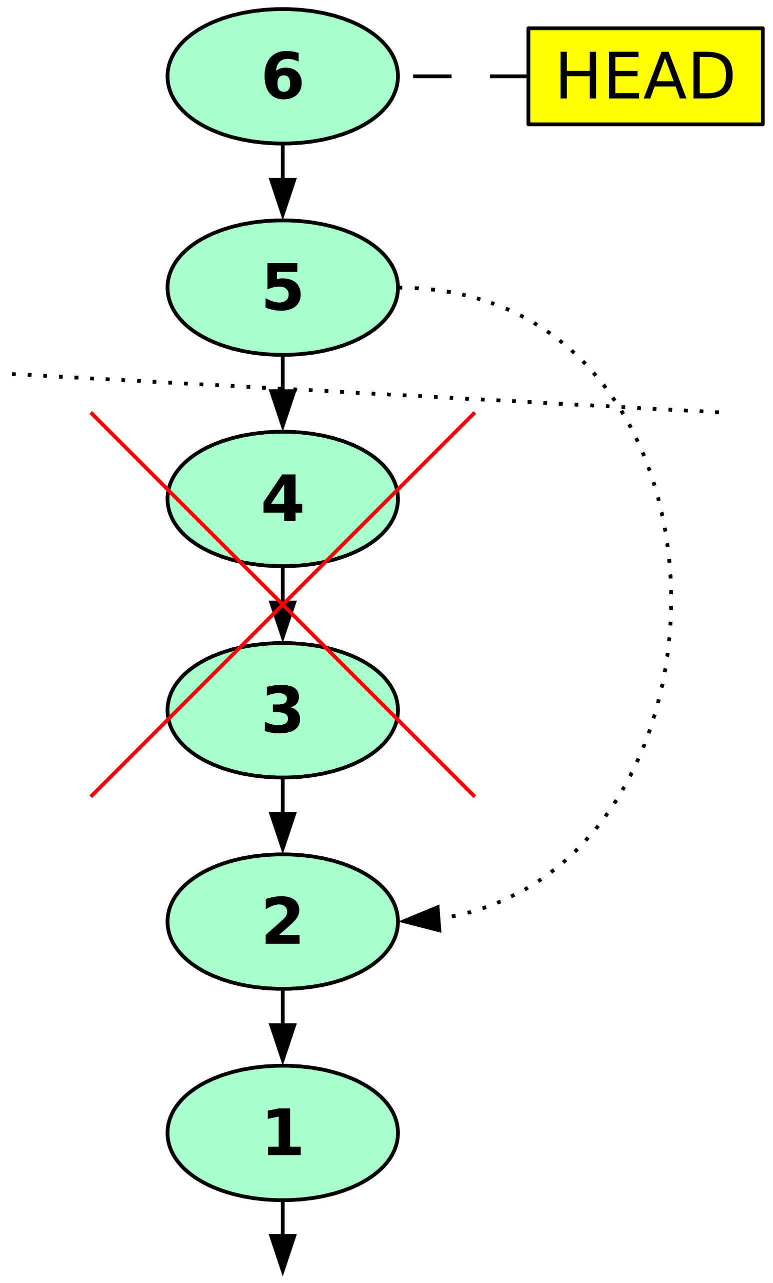
git-rebase --onto <new> [start]



git-rebase --onto <new> [start]

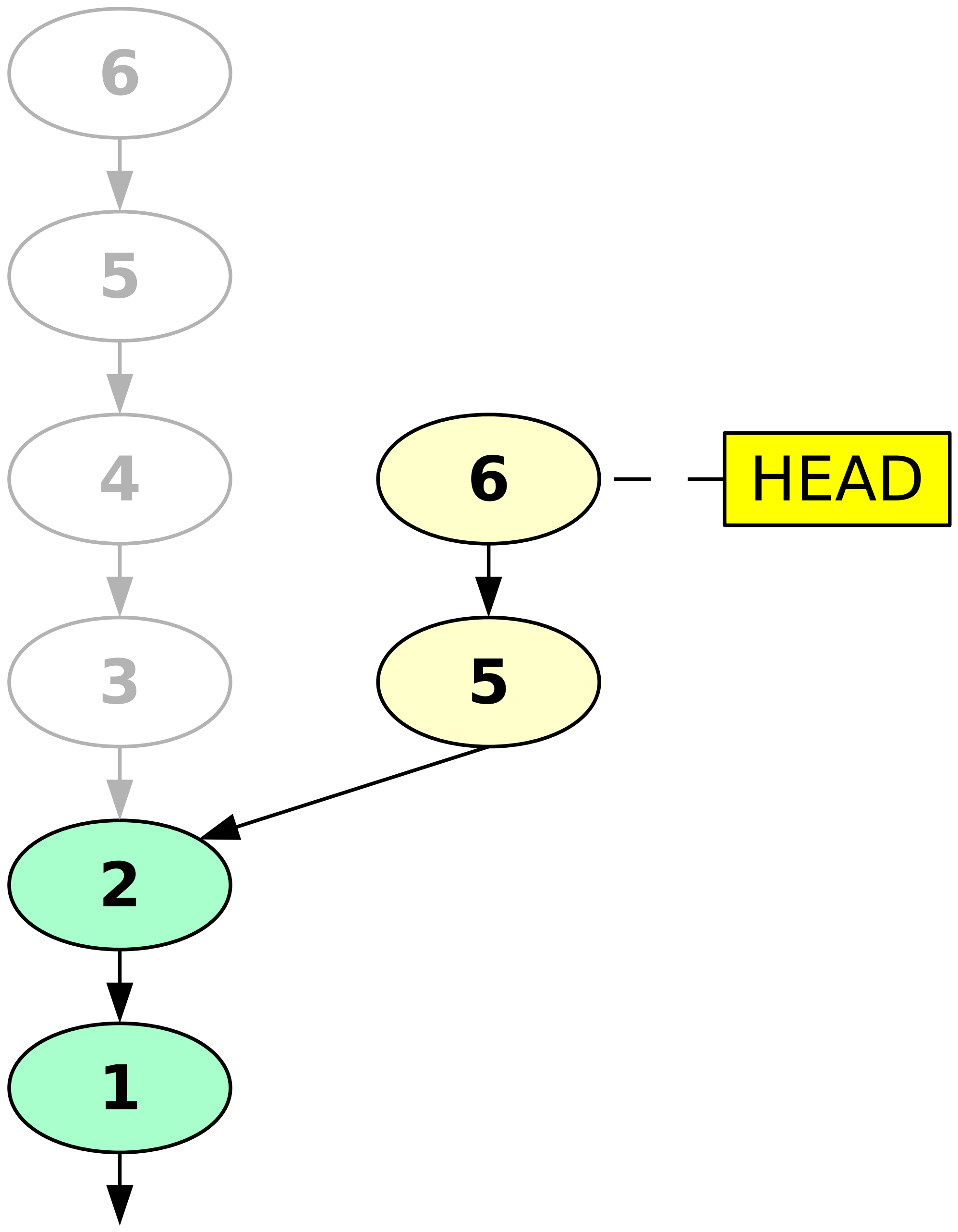


git-rebase --onto <new> [start]



```
git rebase --onto HEAD~4 HEAD~2
```

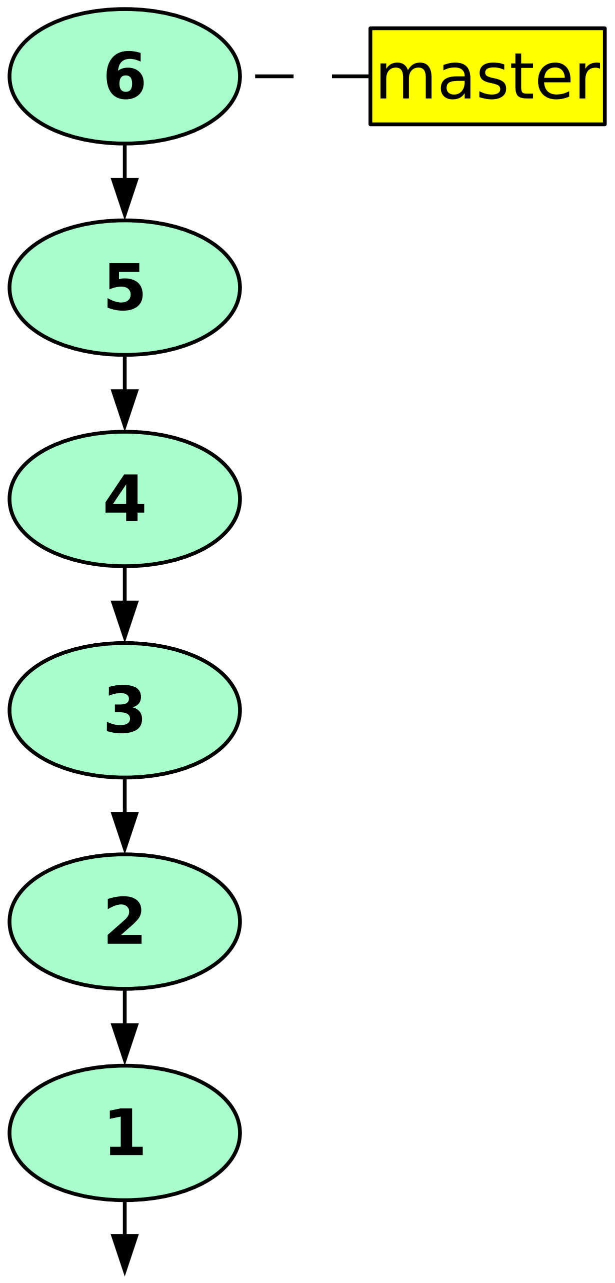
git-rebase --onto <new> [start]



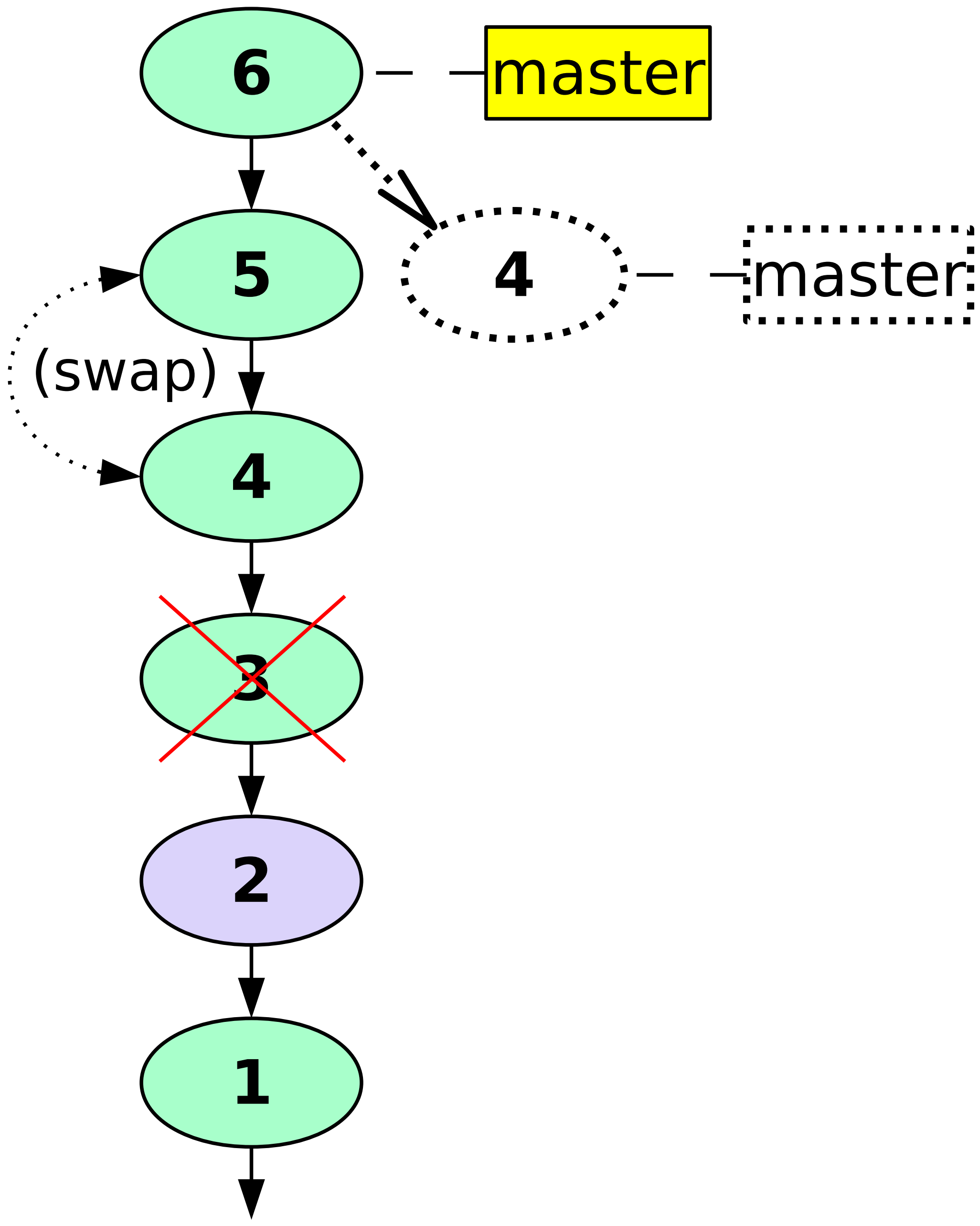
git-rebase

- use cases
  - git rebase [upstream]
    - rebase current branch to "upstream" branch
  - git rebase --onto <new> [start]
    - take all commits from "start" up to the current branch, rebase them on top of "new" commit/branch
  - git rebase -i <refspec>
    - does an interactive rebase of the current branch against "refspec", which should be in the history of the current branch, like "HEAD~5"
  - git rebase --continue | --abort | --skip
    - useful when rebase is interrupted (ie. conflicts)

git-rebase -i <refspec>

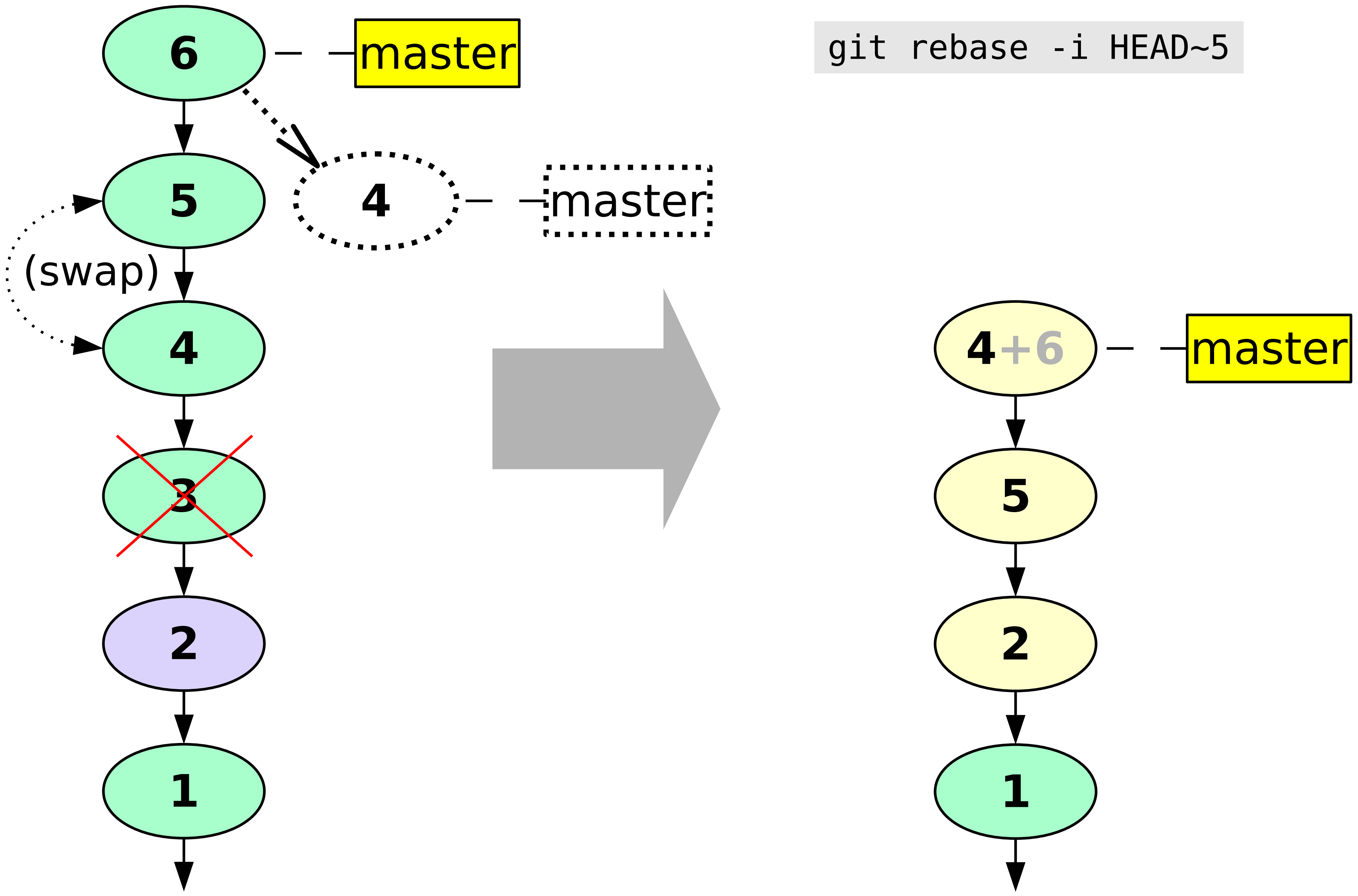


git-rebase -i <refspec>



git-rebase -i <refspec>

```
git rebase -i HEAD~5
```



git-rebase -i <refspec>

```
git rebase -i HEAD~5
```

```
pick c6d44b4 second commit
pick 64e59ca third commit
pick 529f734 fourth commit
pick b9512a8 fifth commit
pick 567653c sixth commit

# Rebase d9e4b6e..567653c onto d9e4b6e
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

git-rebase -i <refspec>

```
git rebase -i HEAD~5
```

```
e c6d44b4 second commit
pick 64e59ca third commit
pick 529f734 fourth commit
pick b9512a8 fifth commit
pick 567653c sixth commit

# Rebase d9e4b6e..567653c onto d9e4b6e
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

git-rebase -i <refspec>

```
git rebase -i HEAD~5
```

```
e c6d44b4 second commit
pick 529f734 fourth commit
pick b9512a8 fifth commit
pick 567653c sixth commit

# Rebase d9e4b6e..567653c onto d9e4b6e
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

git-rebase -i <refspec>

```
git rebase -i HEAD~5
```

```
e c6d44b4 second commit
pick b9512a8 fifth commit
pick 529f734 fourth commit
pick 567653c sixth commit

# Rebase d9e4b6e..567653c onto d9e4b6e
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

git-rebase -i <refspec>

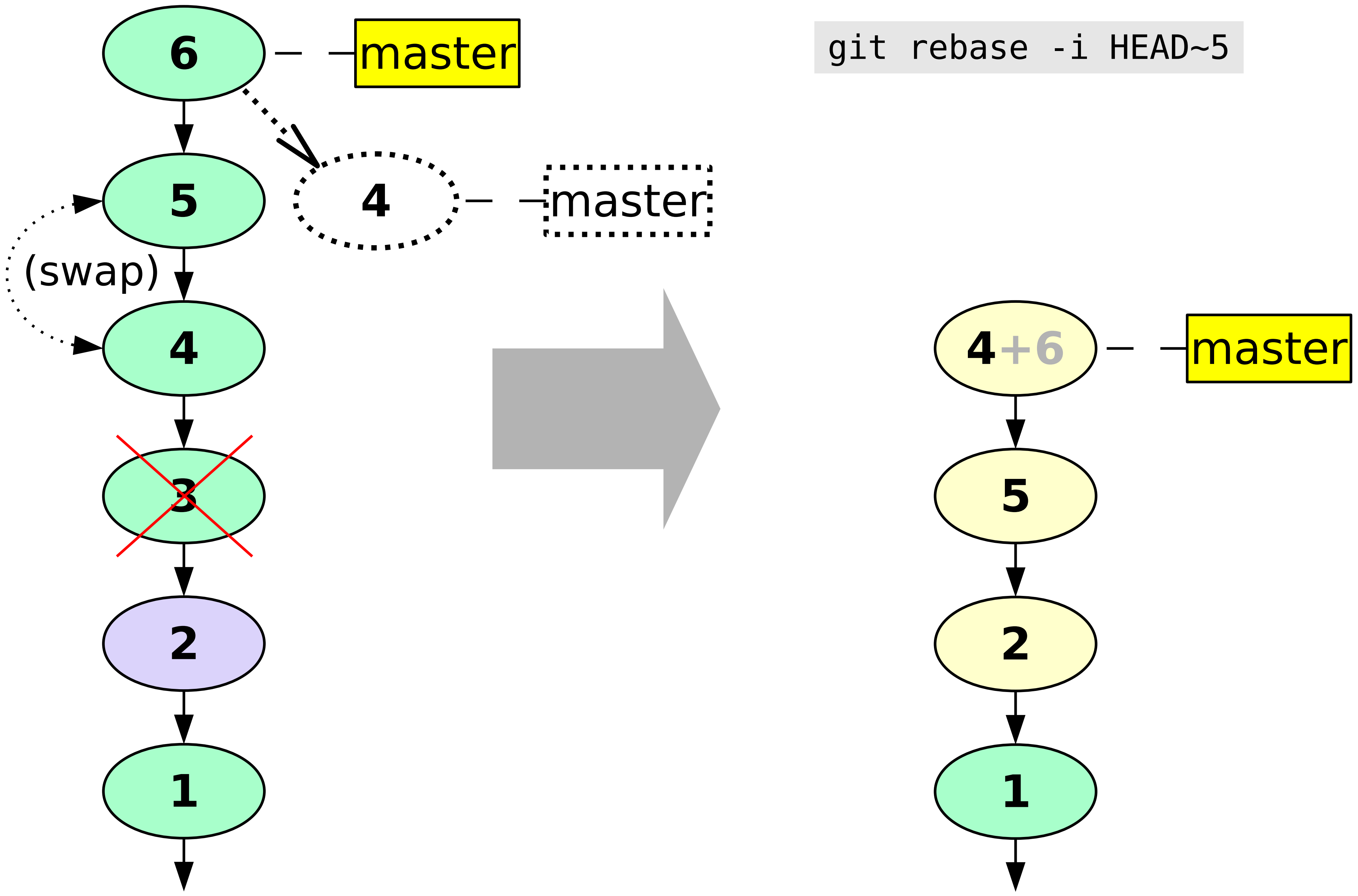
```
git rebase -i HEAD~5
```

```
e c6d44b4 second commit
pick b9512a8 fifth commit
pick 529f734 fourth commit
s 567653c sixth commit

# Rebase d9e4b6e..567653c onto d9e4b6e
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

git-rebase -i <refspec>

```
git rebase -i HEAD~5
```



git-rebase -i <refspec>

```
git rebase -i HEAD~5
```

```
pick 2e318d9 this super cool feature
f 26f6c9c forgot typo
f a2b5e39 damn the semicolon
f 1f797cd comment out all safety checks
f a196116 add /*magic*/ comments to hard-to-understand parts

# Rebase 13d9a94..a196116 onto 13d9a94
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

Summary

Summary

- Rebase = ?
- Fast-forward "merge"
- Cherry picking
- git-rebase
  - git rebase [upstream]
  - git rebase --onto <new> [start]
  - git rebase -i <refspec>

Presentation:
Questions?

Workshop >>

`use workshop-04 repository/branch for this workshop`

Workshop

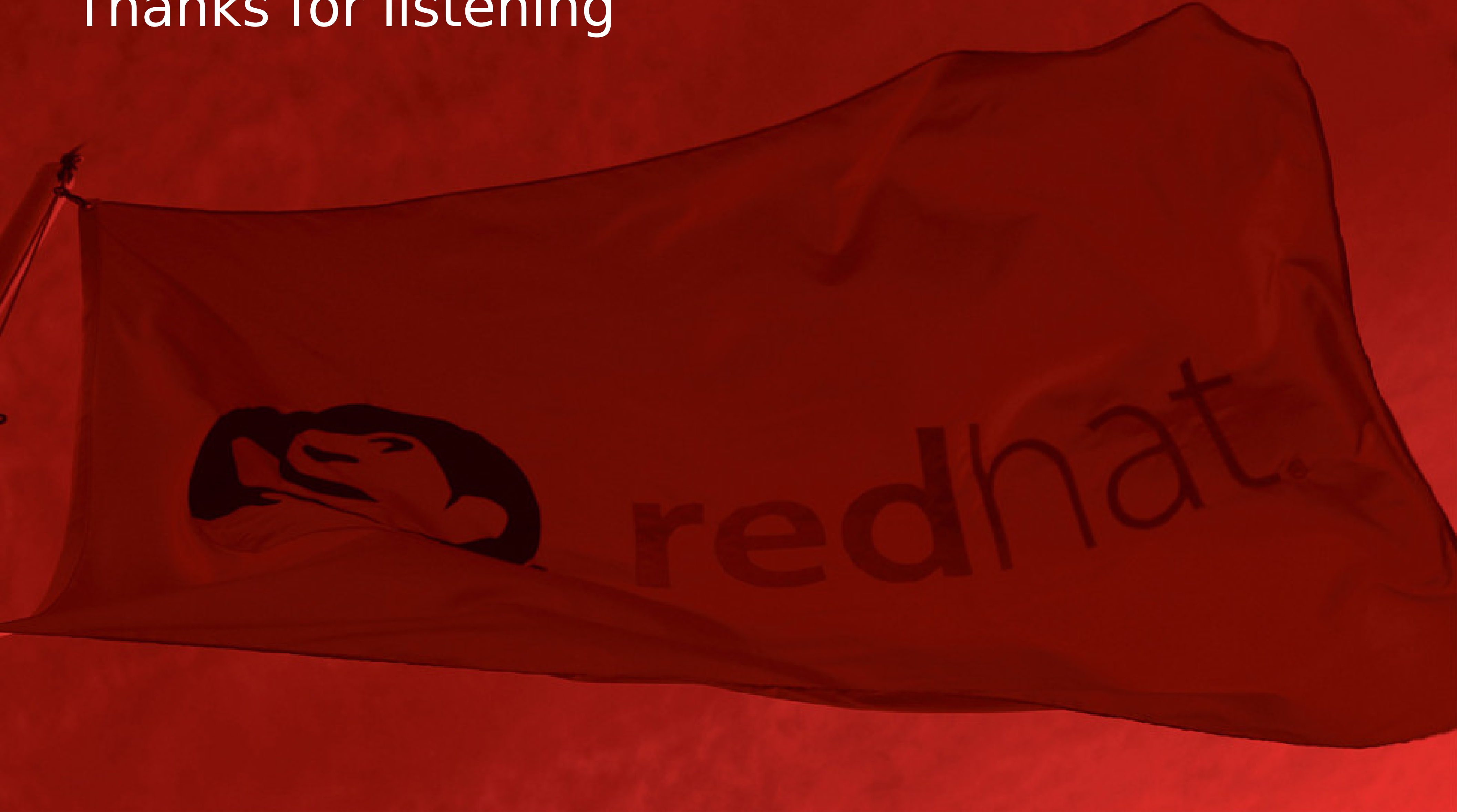
goal: **clean up and rebase a branch**, using interactive rebase, and rebase [upstream] or --onto

1) **See changes**
   - `git log --oneline --graph --decorate --all`
   - gitk
2) **Clean up the repository**
   - identify important changes
   - use interactive rebase - meld fixups, remove debug commits, reword bad commit messages
3) **Rebase against origin/master**
   - "Initial commit" is the common ancestor
4) **Result**
   - something like

```
9f2eacf (HEAD, mkdoc) add html-based documentation
4dbdfce add command lists for all relevant commands
f3305f4 add manpage-based text documentation
3e4d0fa (origin/master) test-path-utils: Fix off by one, found by valgrind
625567f chain kill signals for cleanup functions
87debce Add is_regex_special()
f28ba82 Change NUL char handling of isspecial()
26523fe Add ctype test
```

The End

Thanks for listening



Links

- Git-scm.com - branching / rebasing

```
http://git-scm.com/book/en/Git-Branching-Rebasing
```

- Manpage with examples!

```
man 1 git-rebase
```