CS 241: Systems Programming
Lecture 34. Advanced Git

Fall 2019

Prof. Stephen Checkoway

Using "branches"

Development and release versions

Trying out new features

Focusing on fixing a bug

Simpler to do in Git than other VCS, consider using more frequently

Branches

Visualize a project’s development as a “linked list” of commits.

When a development track splits, a new branch is created.

In Git, branches are actually just a pointer to these commits

Git branching

List all branches in the project

- `git branch`

Create a new branch

- `git branch <branchname>`

Switch to a branch

- `git checkout <branchname>`

Create and immediately switch

- `git checkout –b <branchname>`

Delete a branch

- `git branch –d <branchname>`

Using branches

Create and switch to a branch

```
$ git branch working

$ git checkout working
M  README
Switched to branch 'working'

$ git branch
  master
* working
```

Stashing

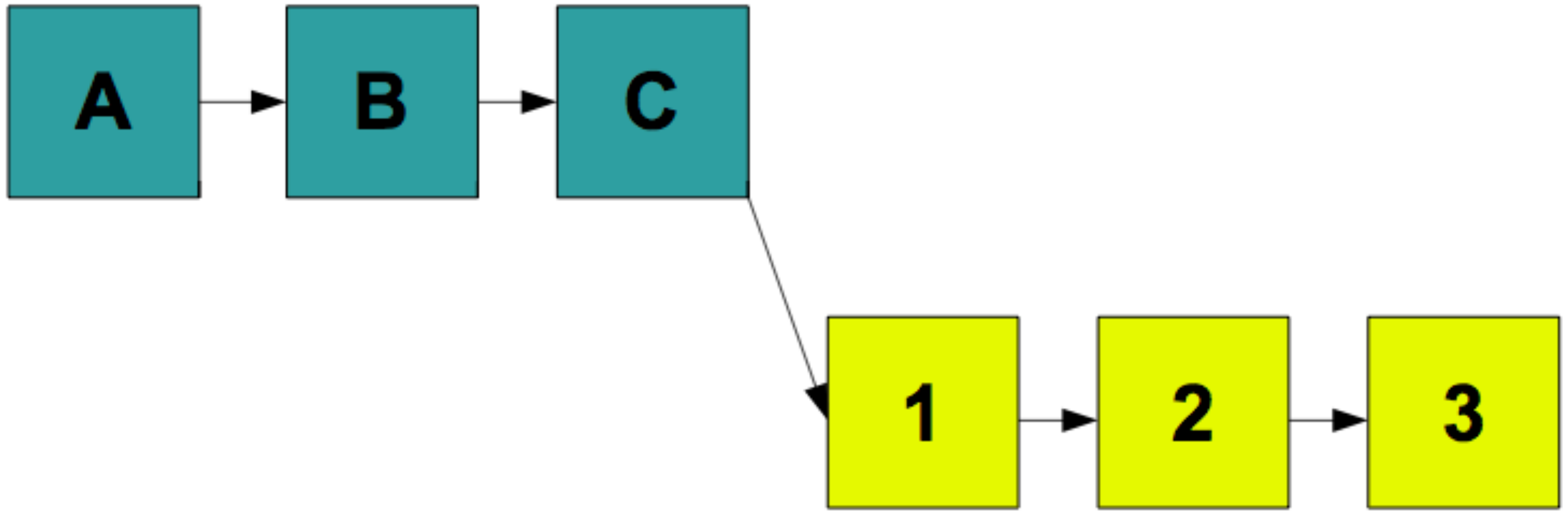
Working tree should be clean when switching branches

Save/hide changes you don't want to commit with `git stash`

- Pushes changes onto a stash stack

Recover changes lager with `git stash pop`

Using branches



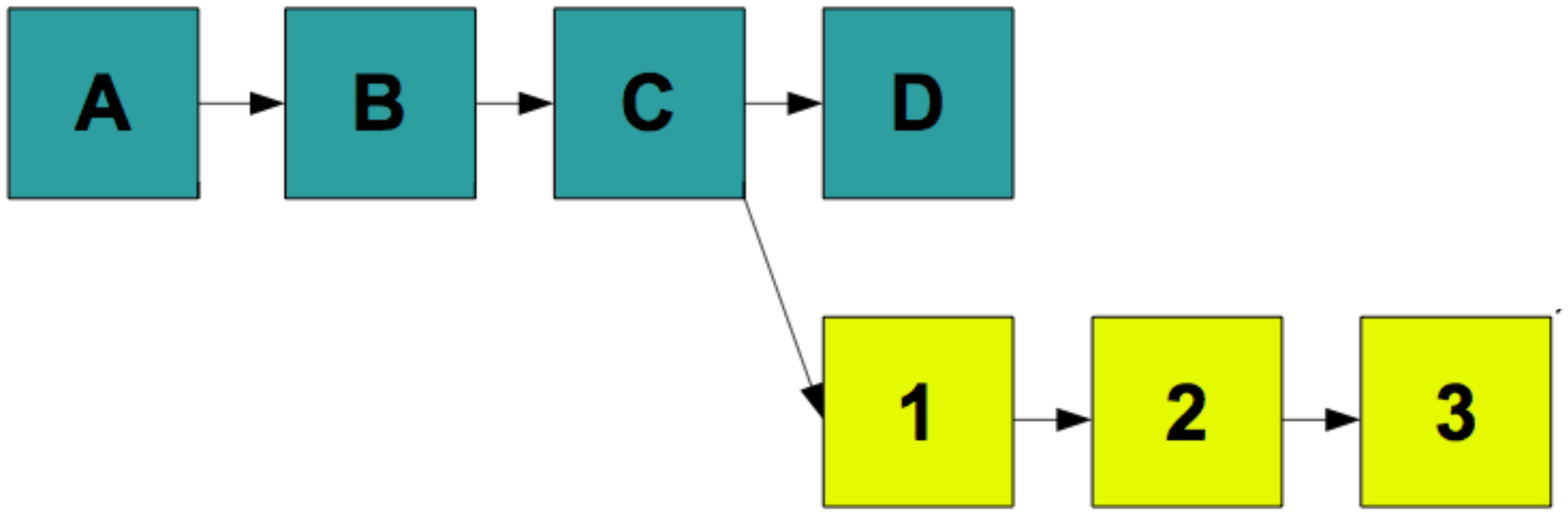
Using branches

Integrate changes back into **master**

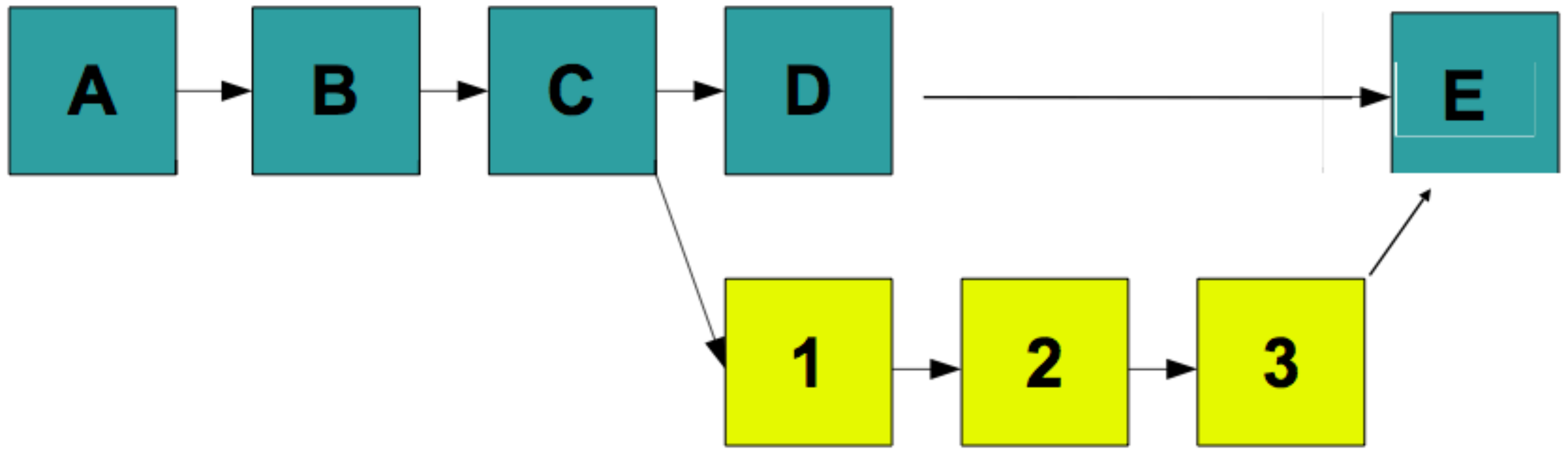
```
$ git checkout master
Switched to branch 'master'

$ git merge working
Merge made by the 'recursive' strategy.
 newfile.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 newfile.txt
```

Before git merge



After git merge



Merged history

```
*   cdd07b2 - (HEAD, master) Merge branch
'working'
|\
| * 1ccf9e7 - (working) Added a new file
* | 3637a76 - Second change
* | cf98d00 - First change
|/
* cf31a23 - Updated README to 2.0
* 2a8fc15 - Initial commit
```

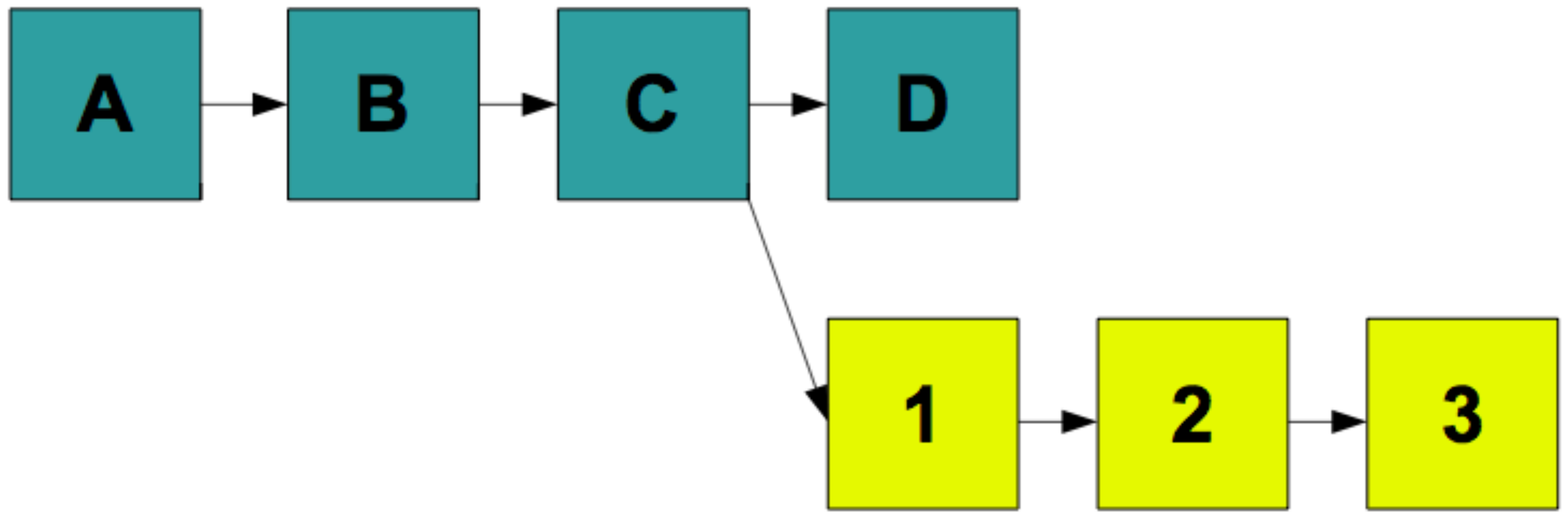
Rebasing

Like merging, rebasing transfers changes from one branch to another

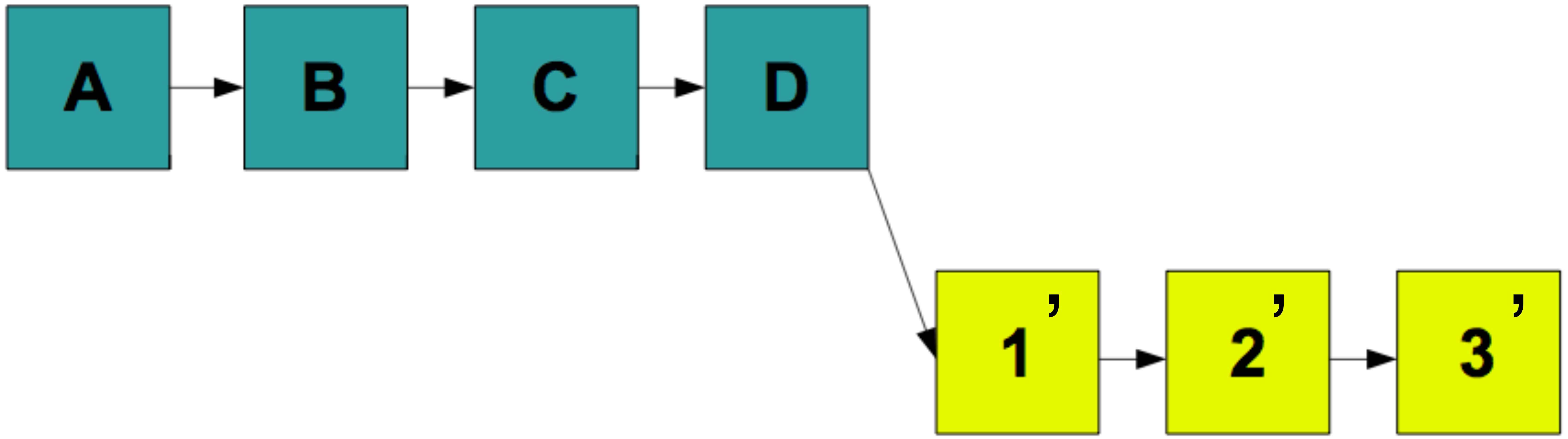
Does not create a new commit

Replays changes from current branch onto head of other branch

Before git rebase



After git rebase



git rebase

Powerful tool

Can change the commit order

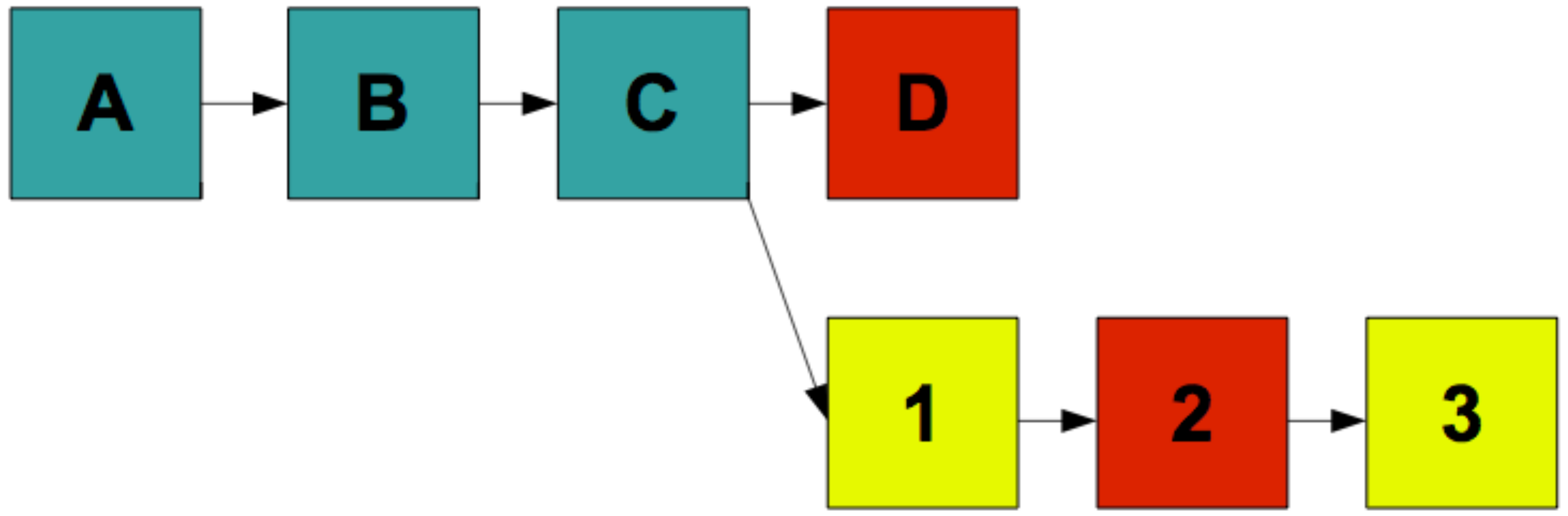
Merge/split commits

Make fixes in earlier commits

- DO NOT DO ON PUSHED CHANGES OR PUBLIC BRANCH

```
$ git rebase –i master
```

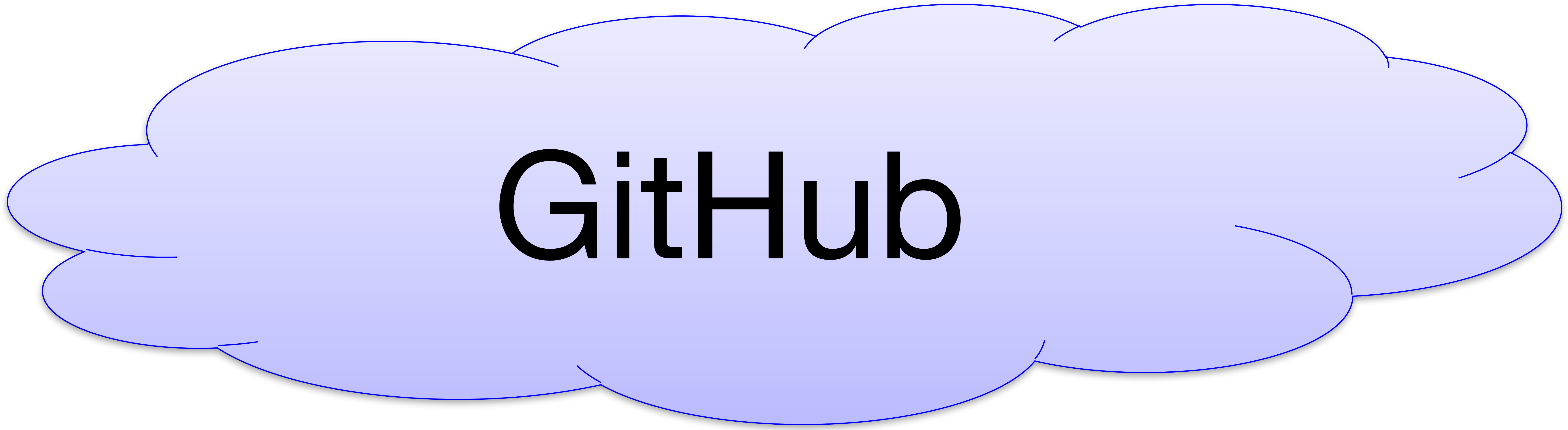
Conflicts



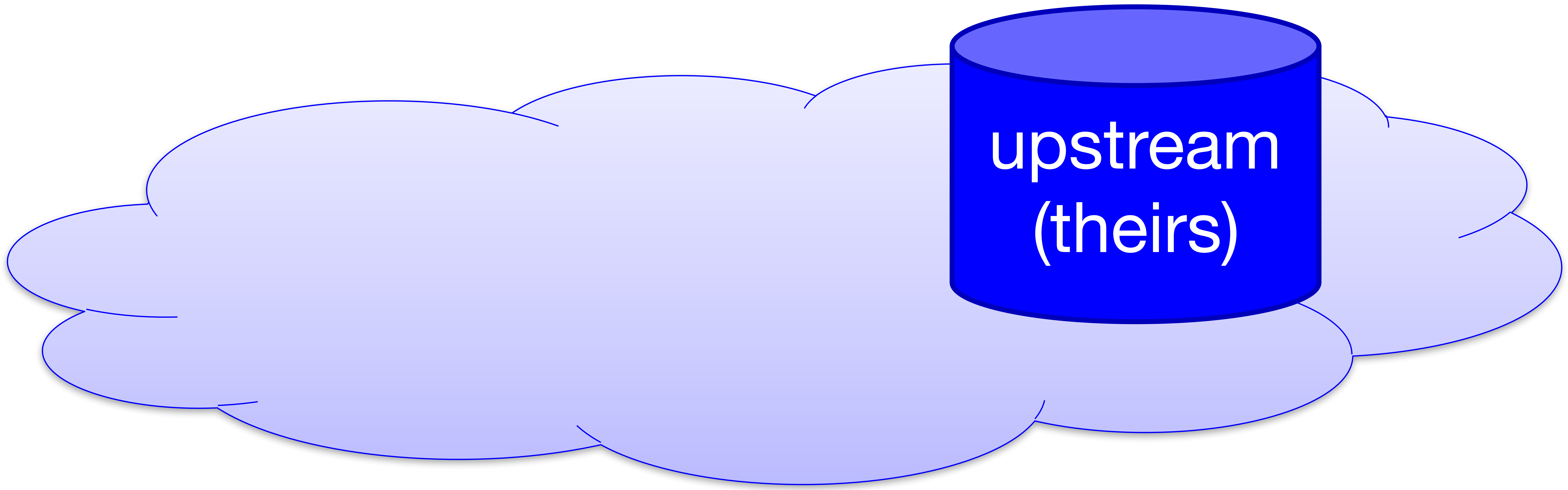
Git conflict markers

```
$ cat foo.c
<<<<<<< HEAD
current content
=======
branch content
>>>>>>> newbranch
$ vim foo.c
$ git add foo.c
$ git rebase --continue
```

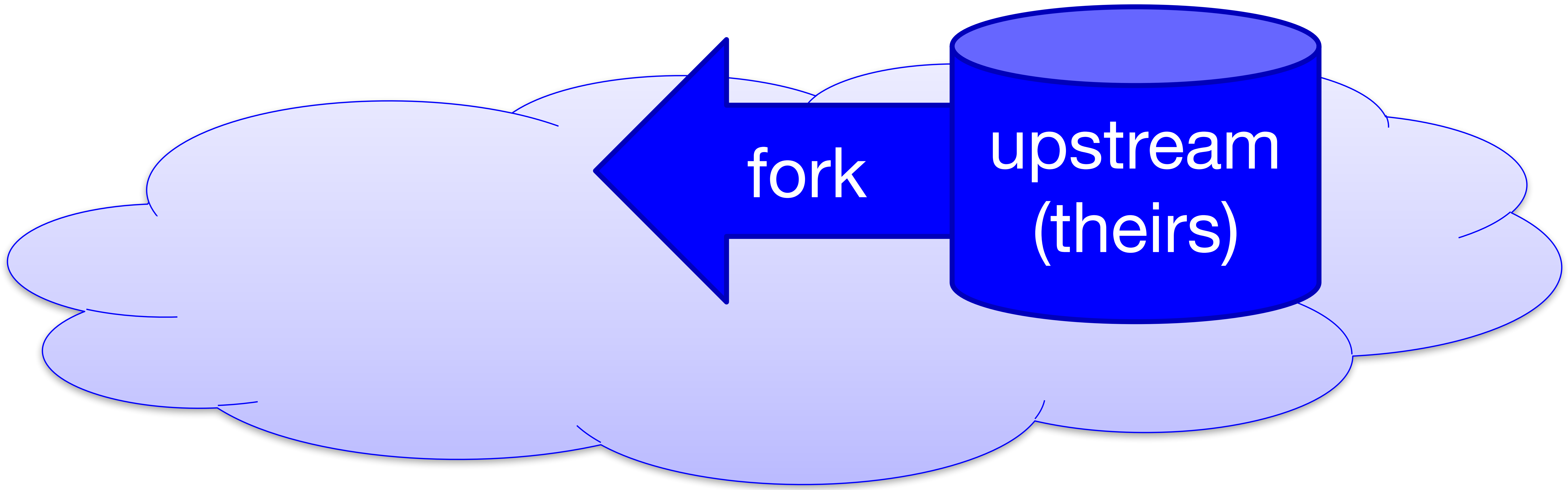
Setup



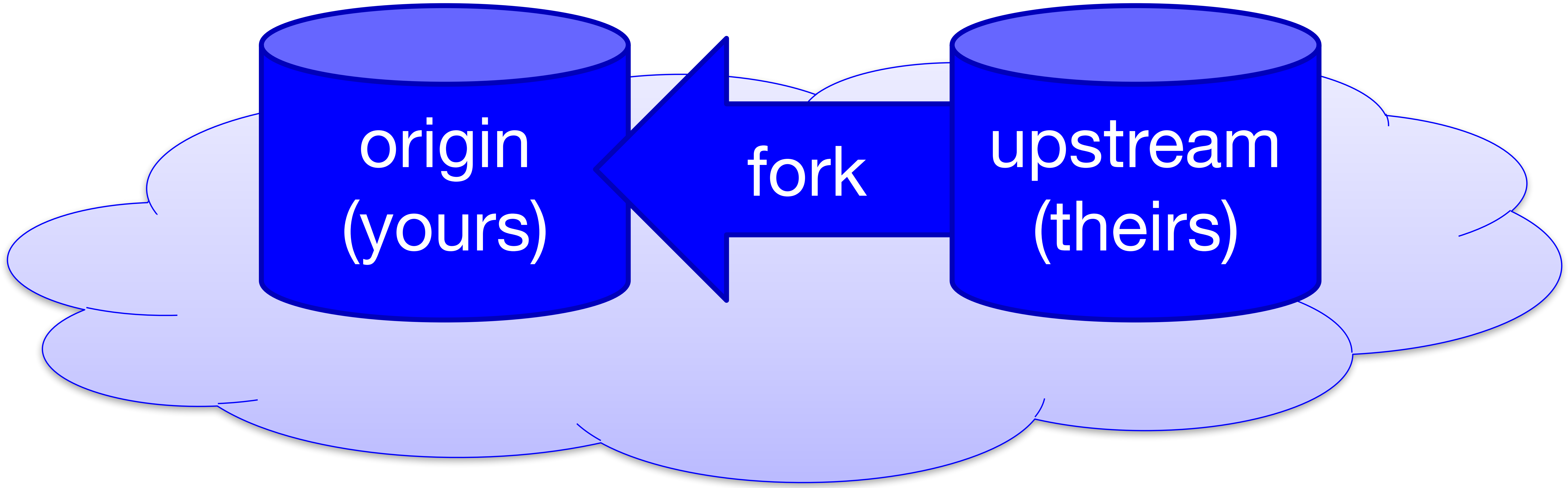
Setup



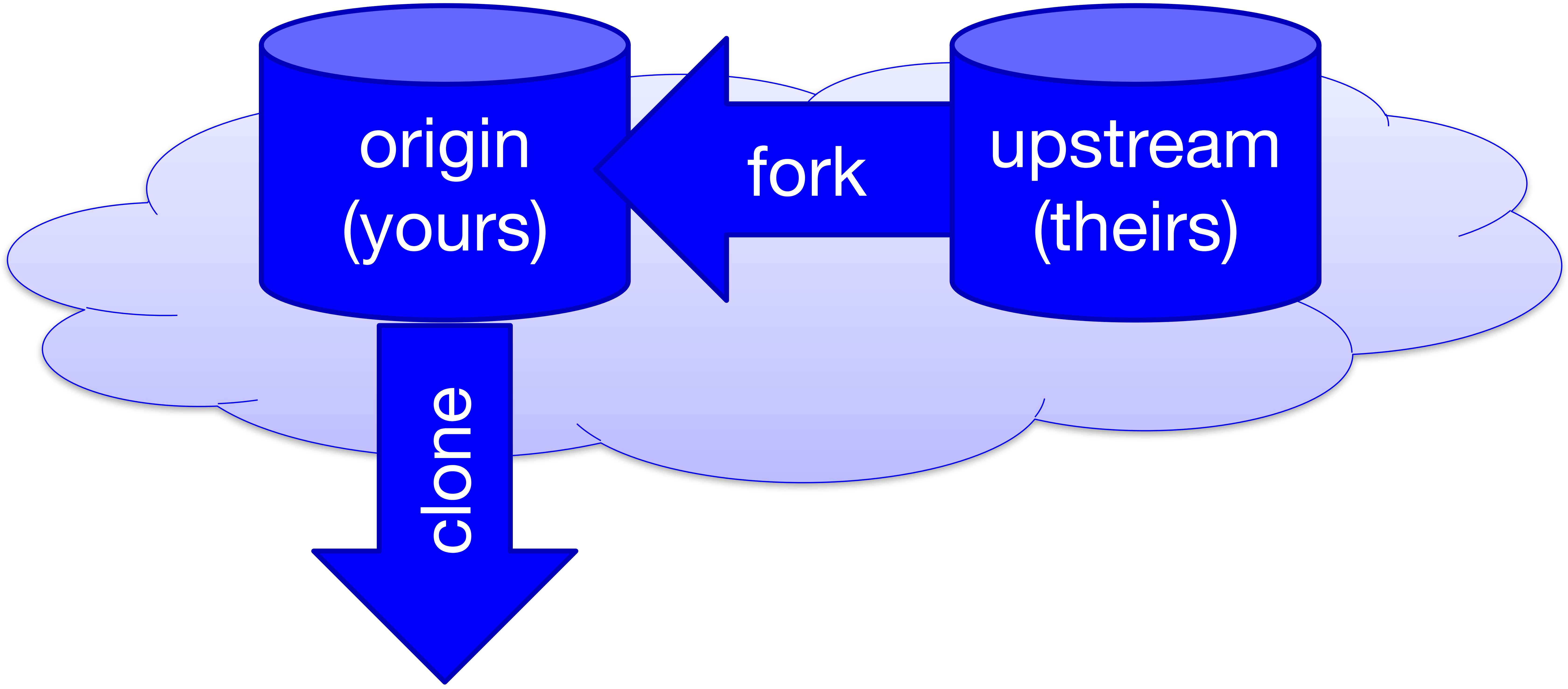
Setup



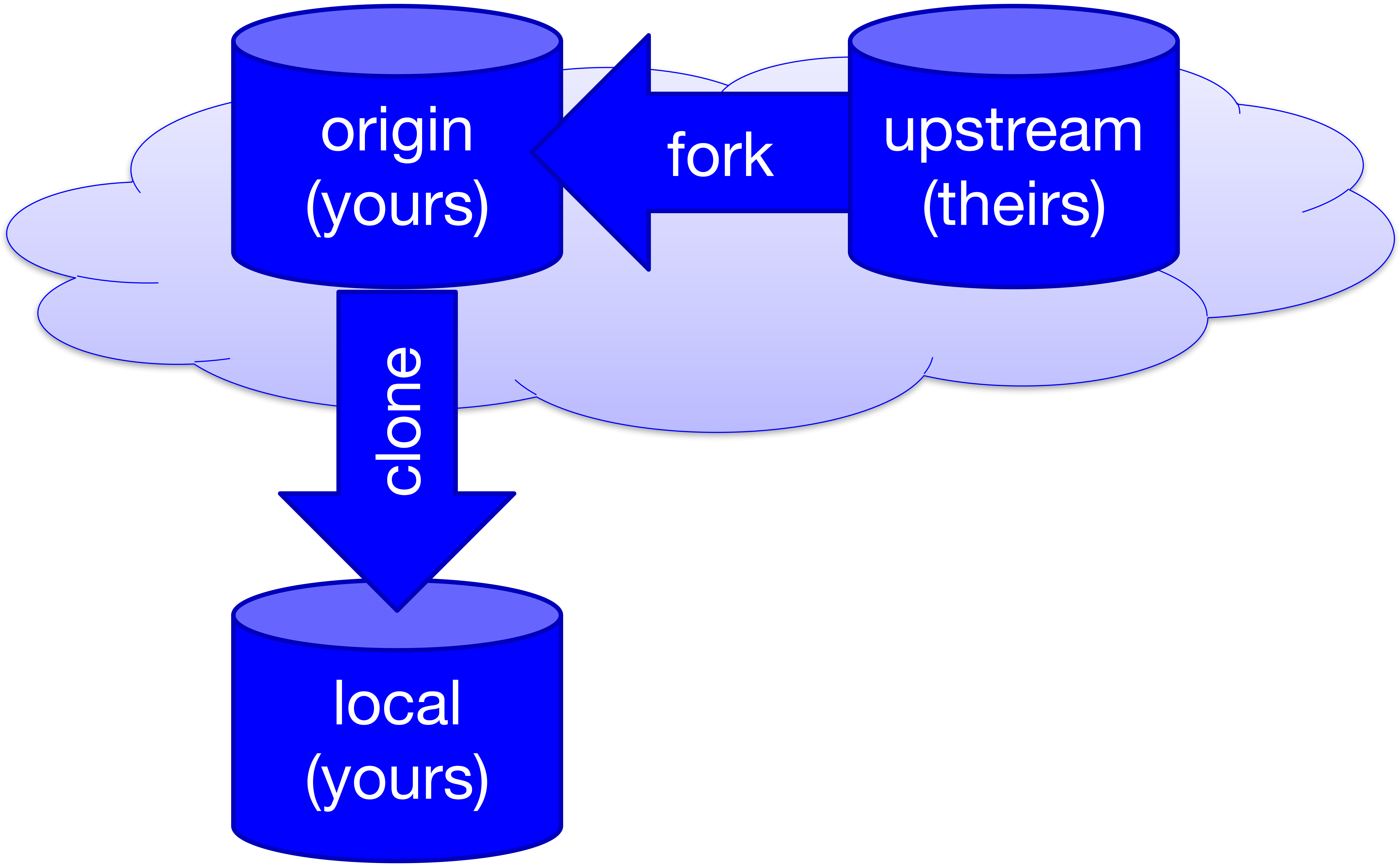
Setup



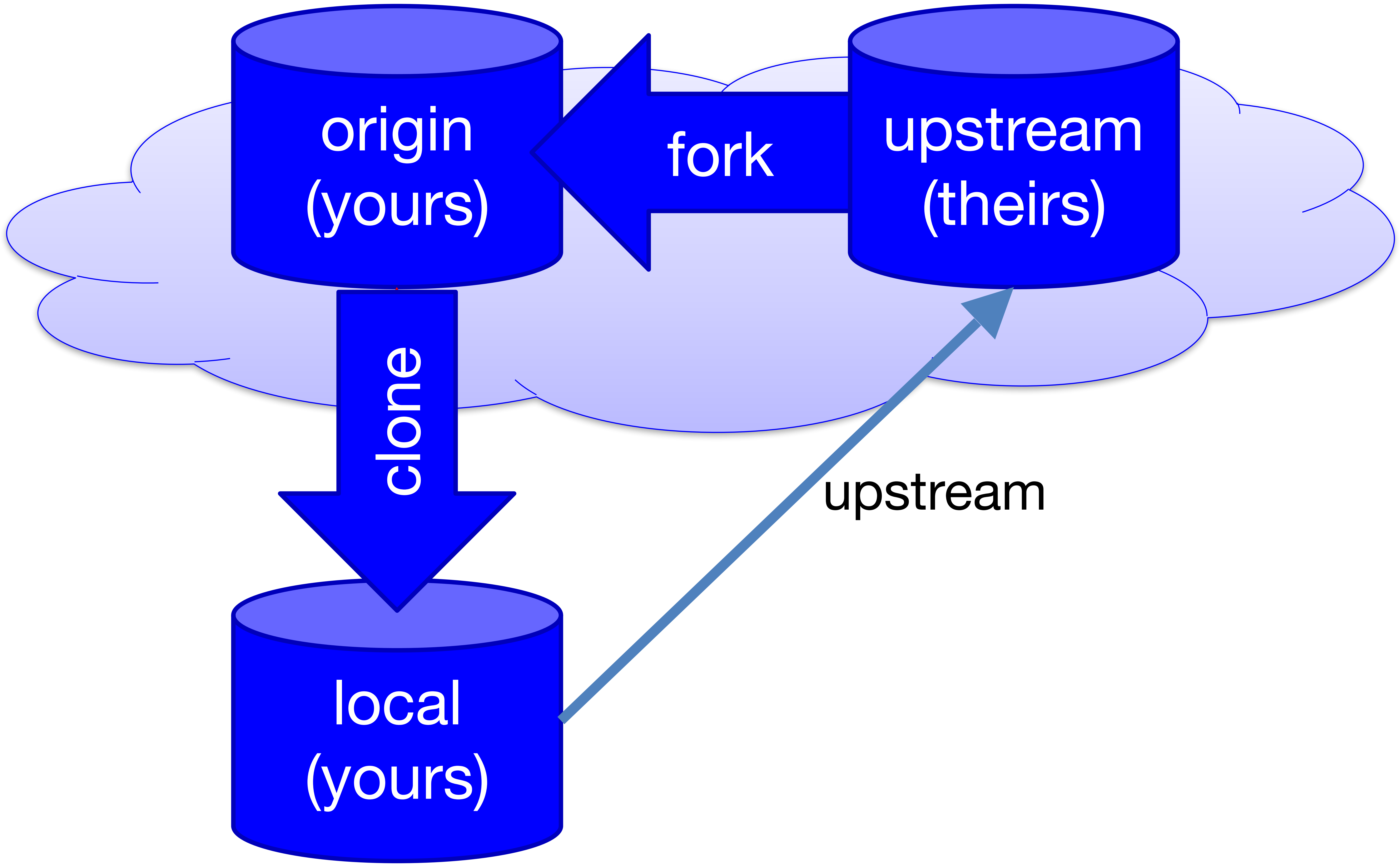
Setup



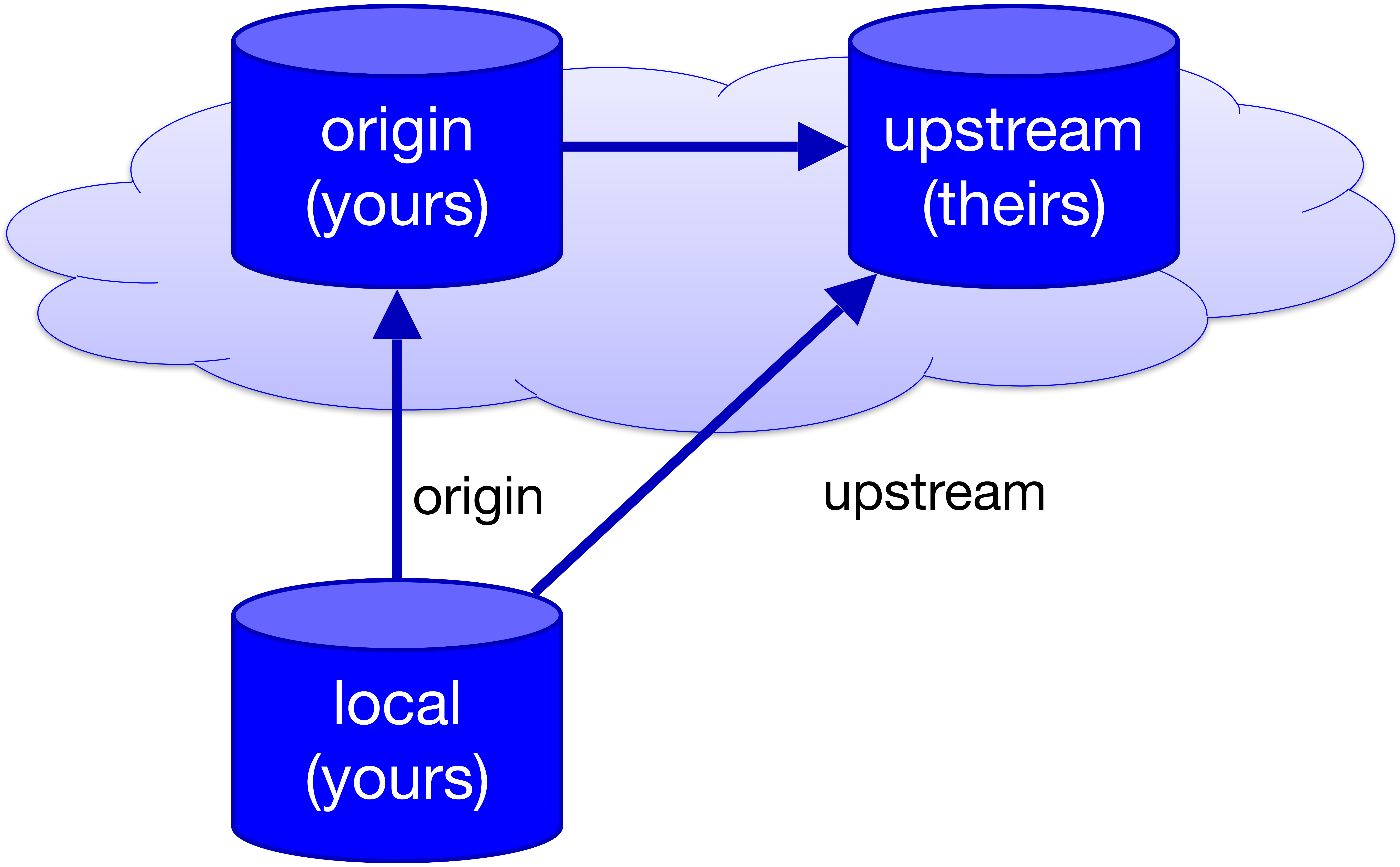
Setup



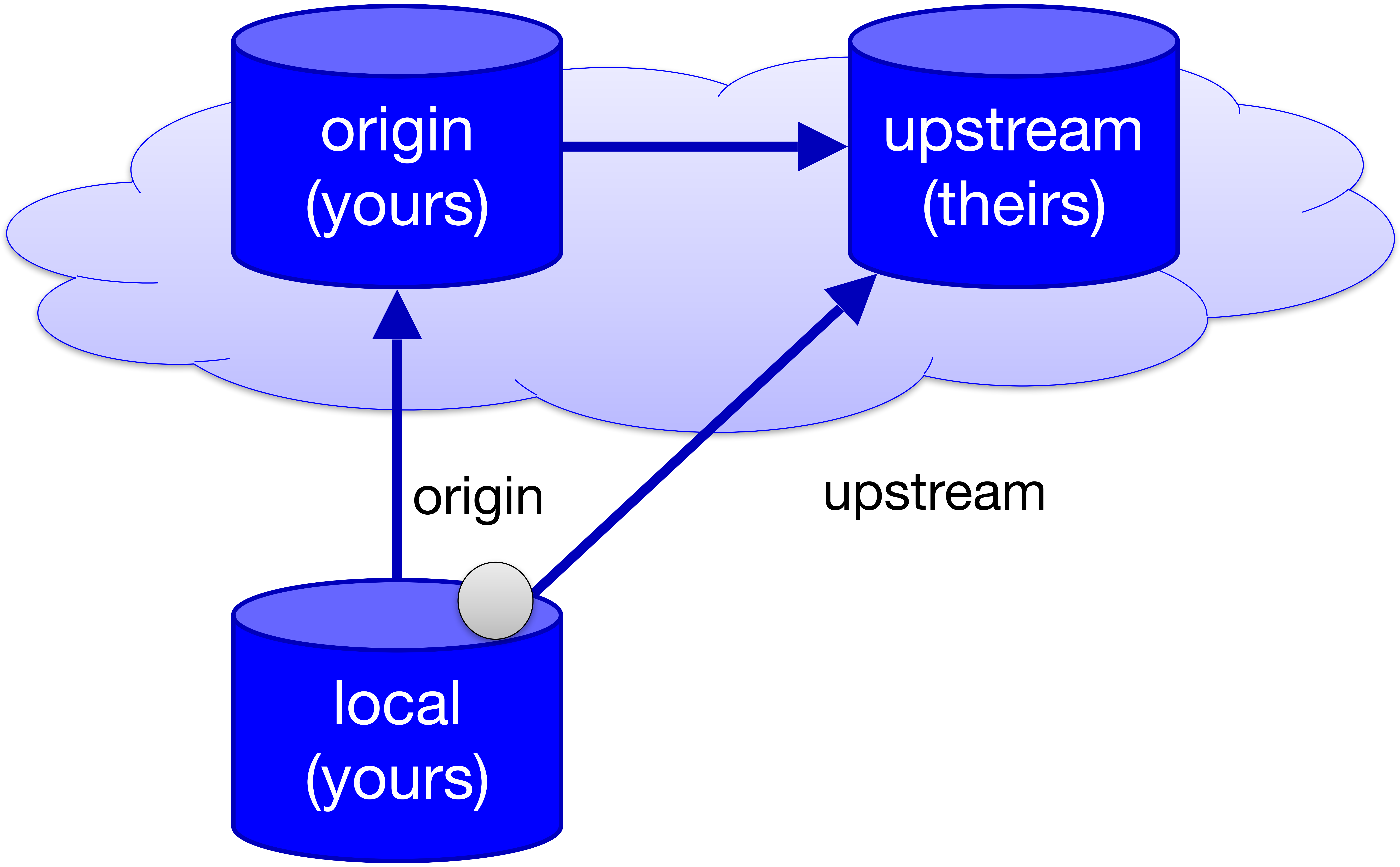
Setup


origin
(yours)
fork
upstream
(theirs)
clone
upstream
local
(yours)

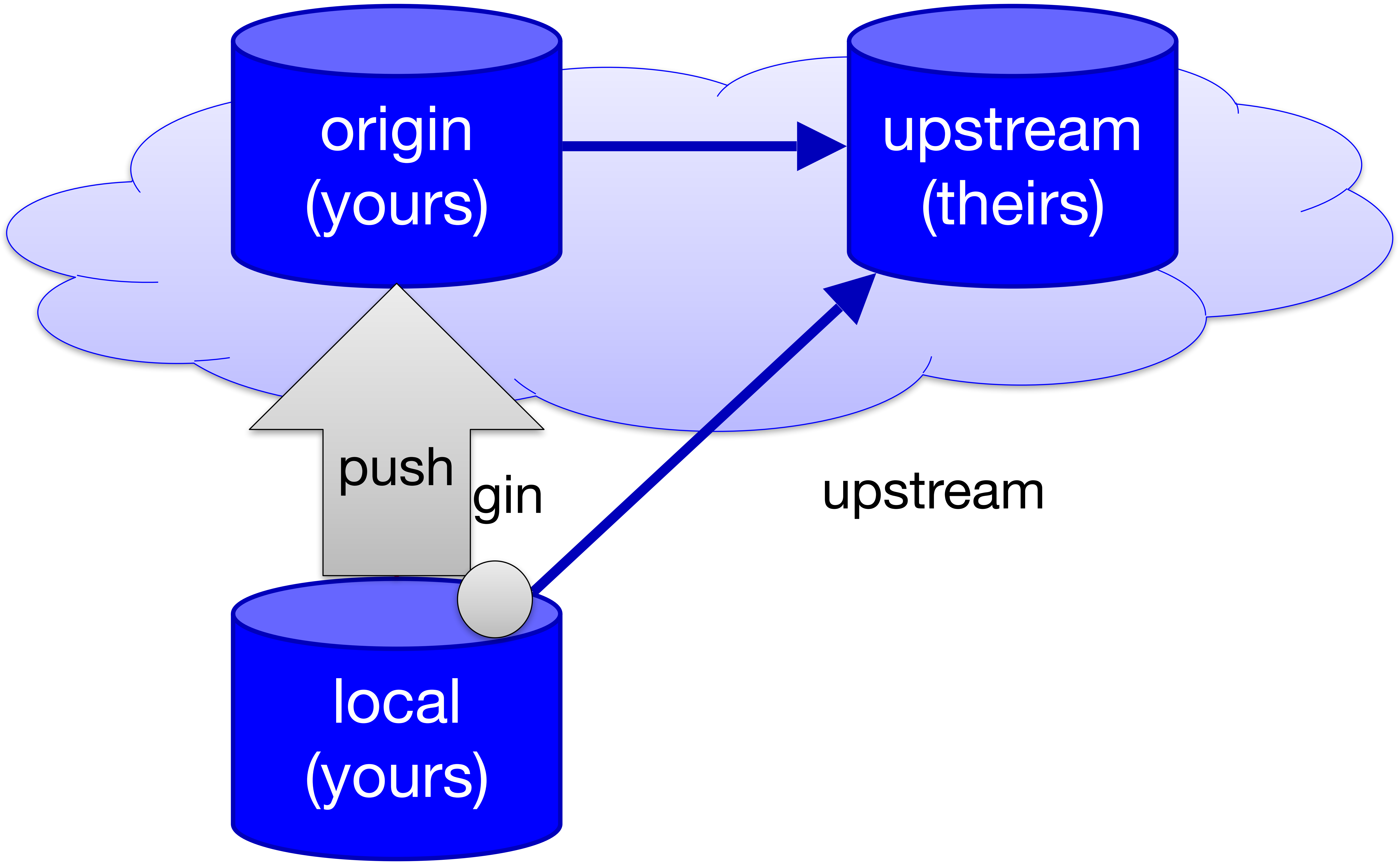
Setup



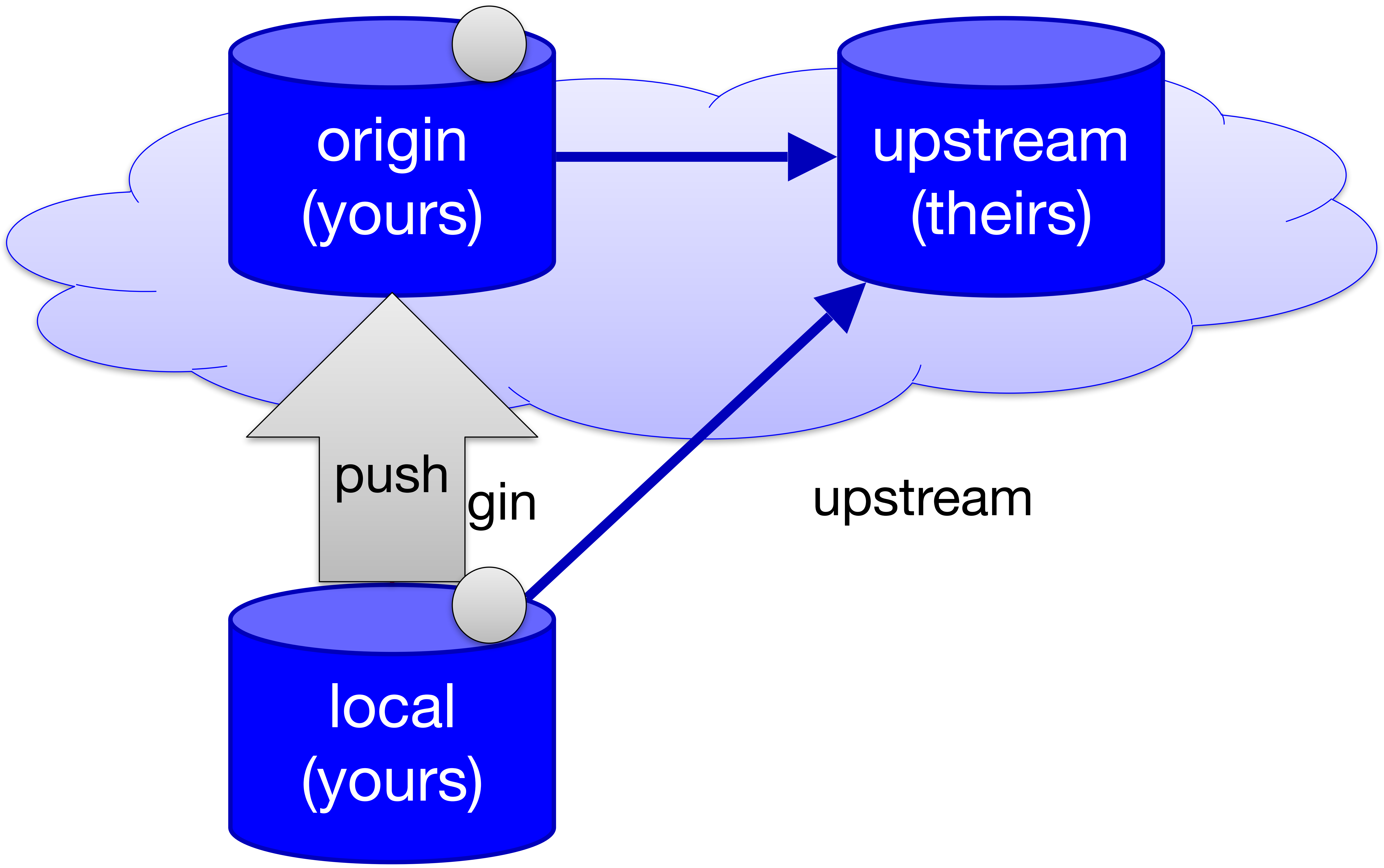
Contribute Changes



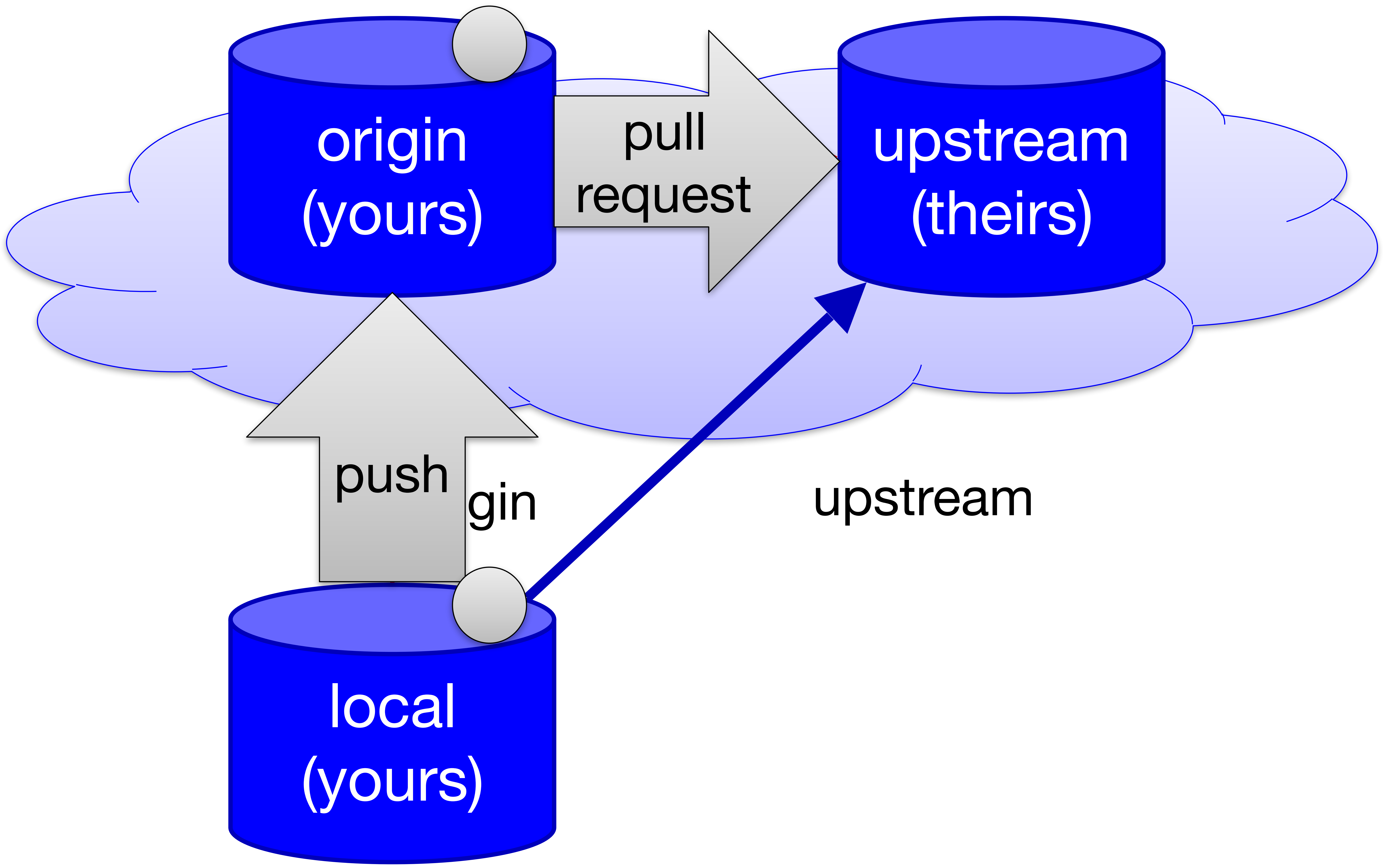
Contribute Changes



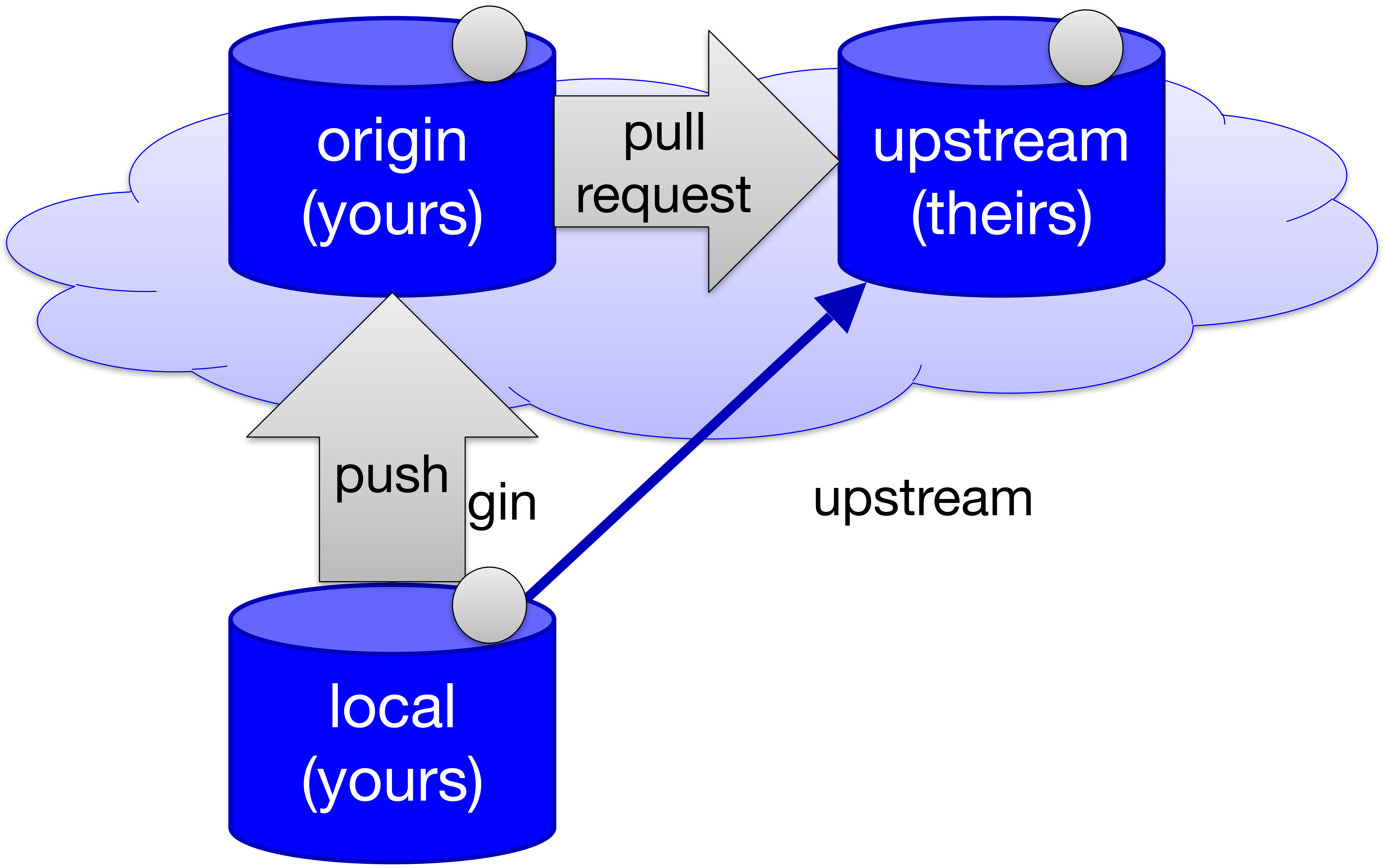
Contribute Changes



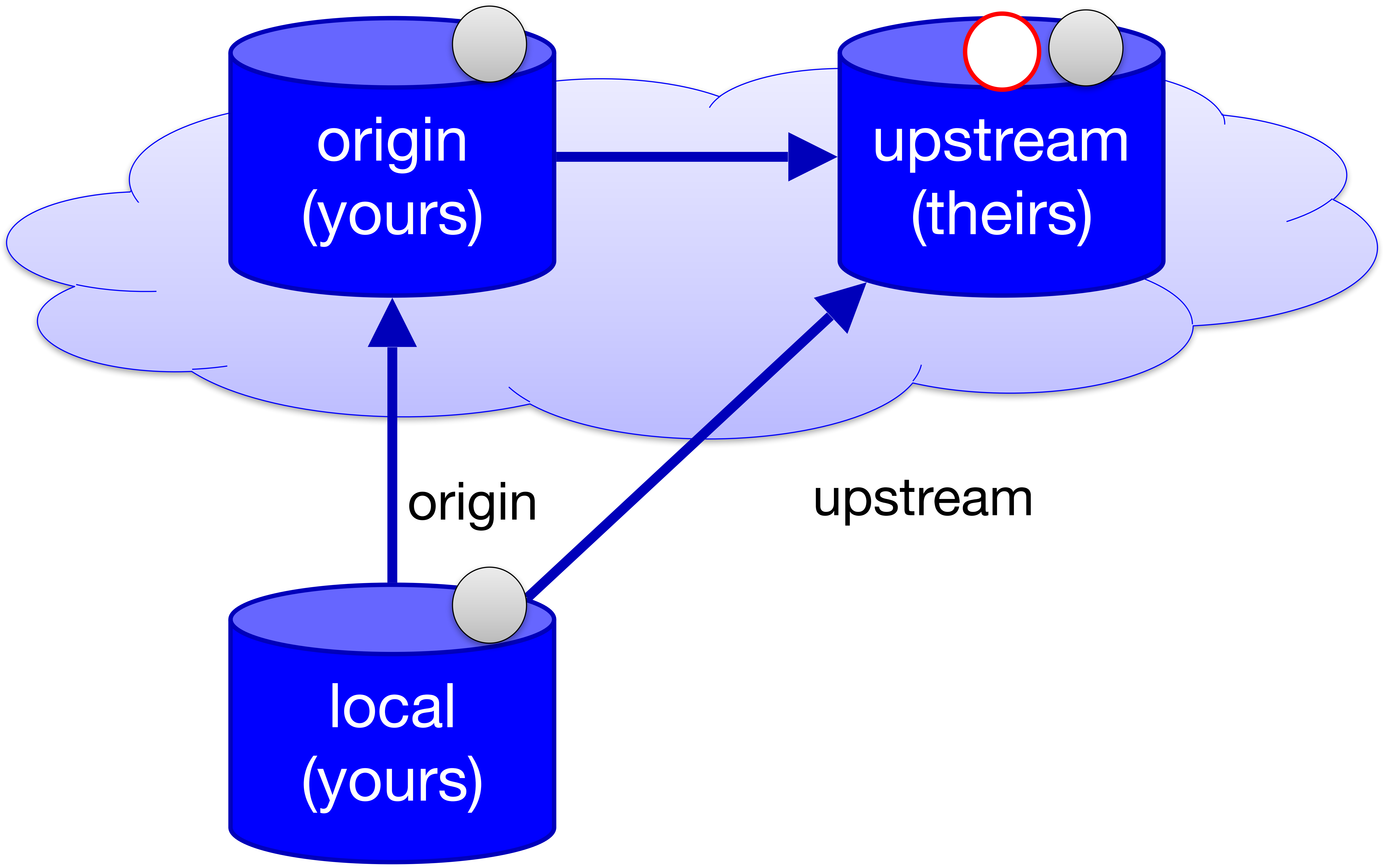
Contribute Changes



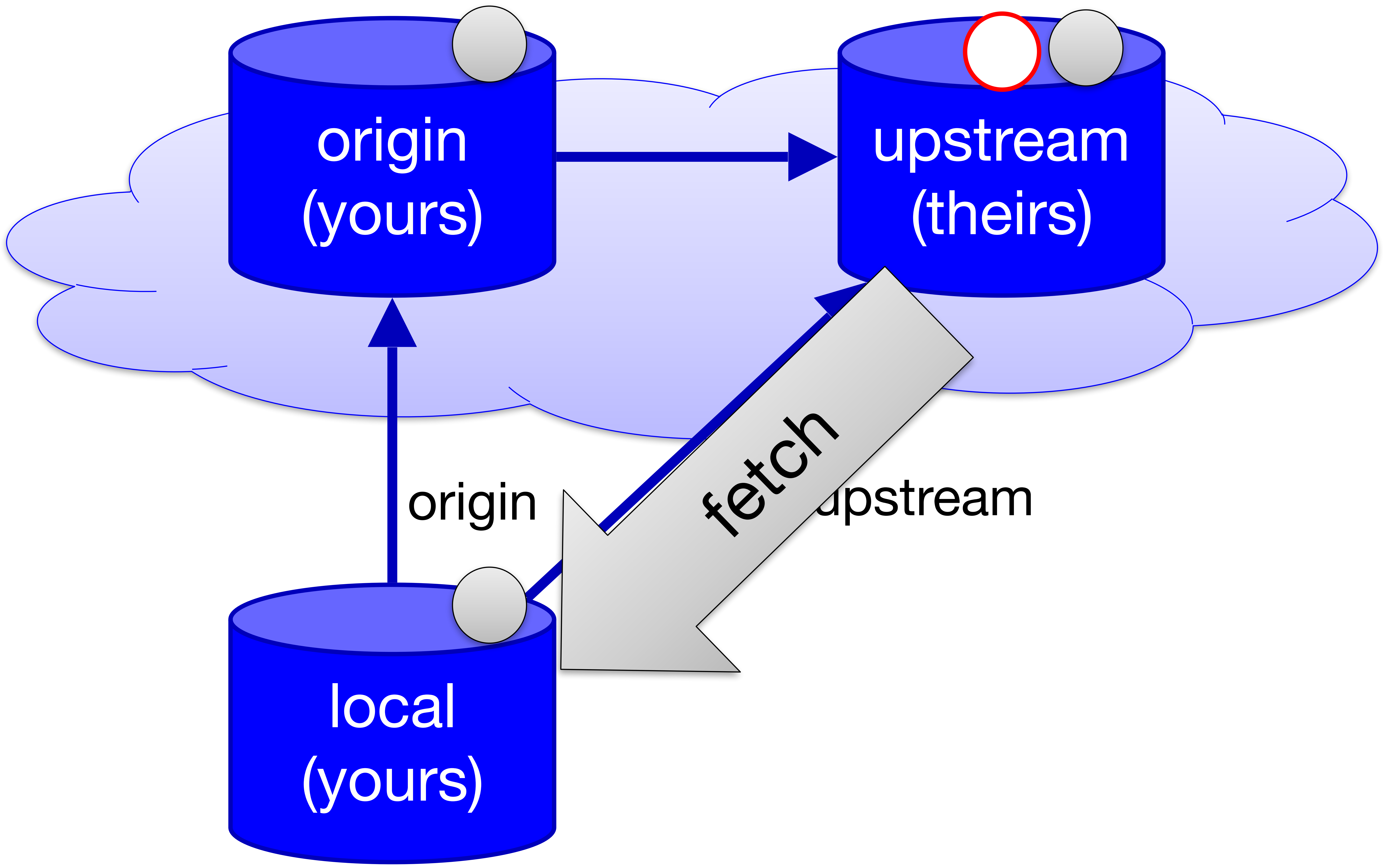
Contribute Changes



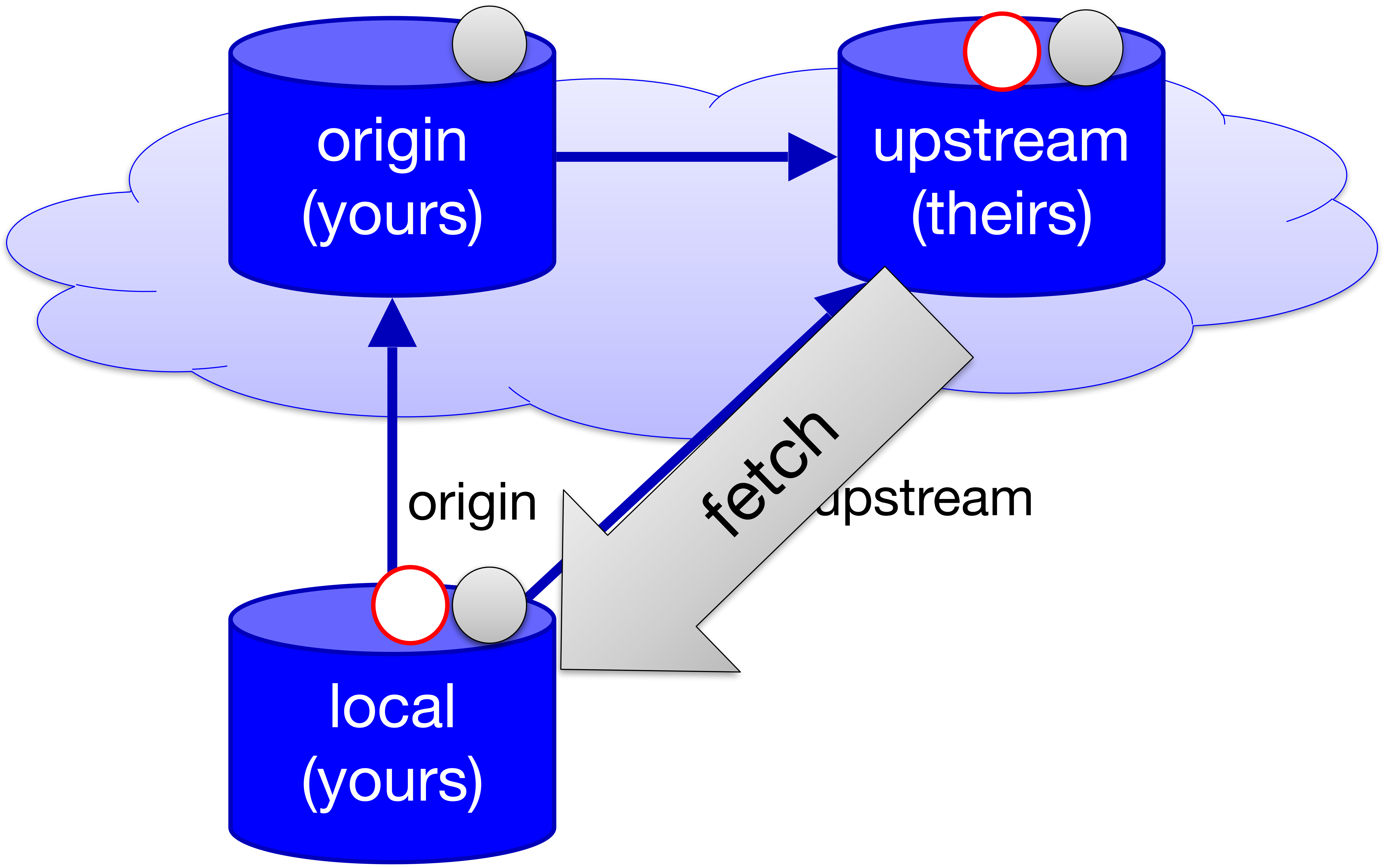
Integrate Changes



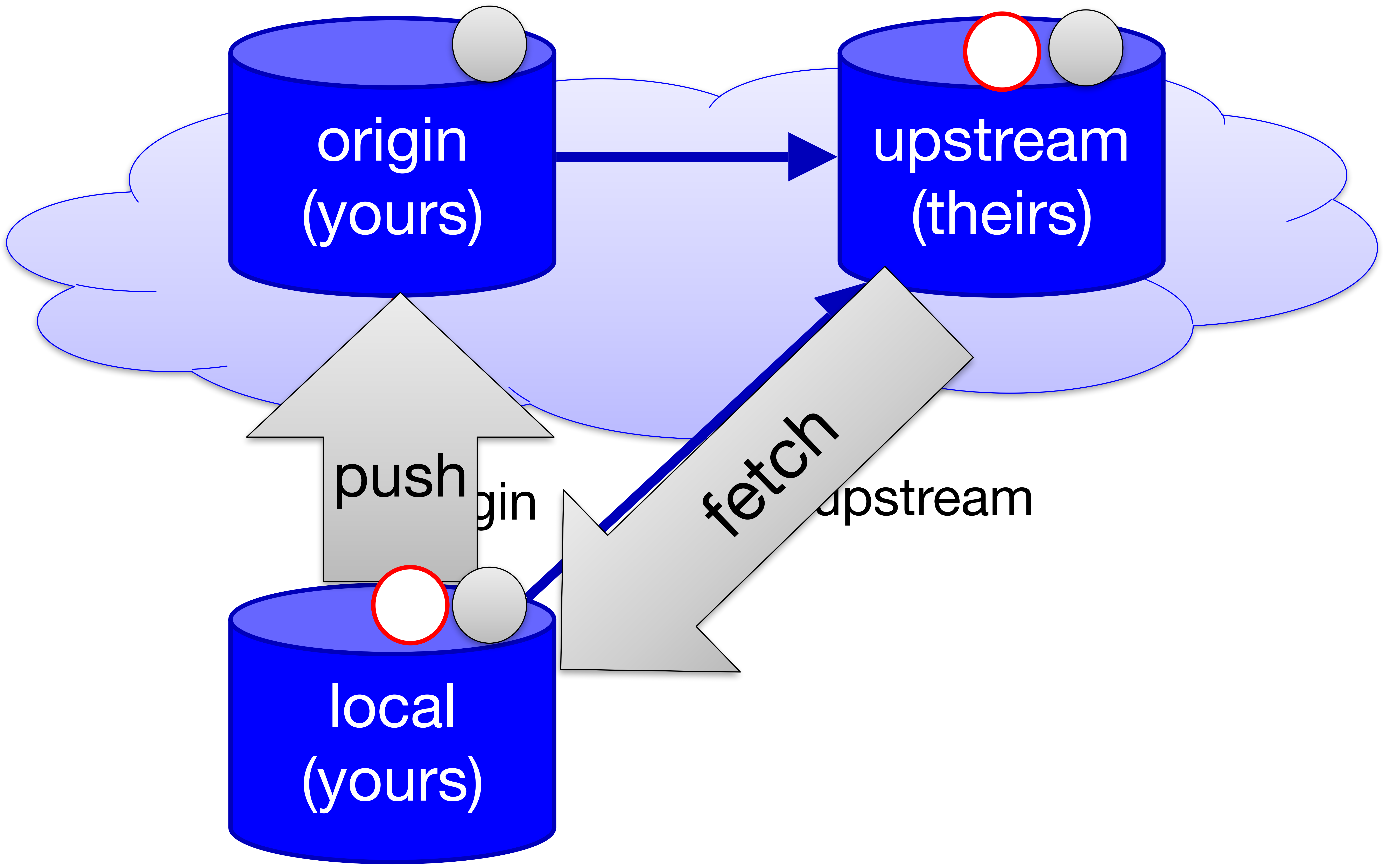
Integrate Changes



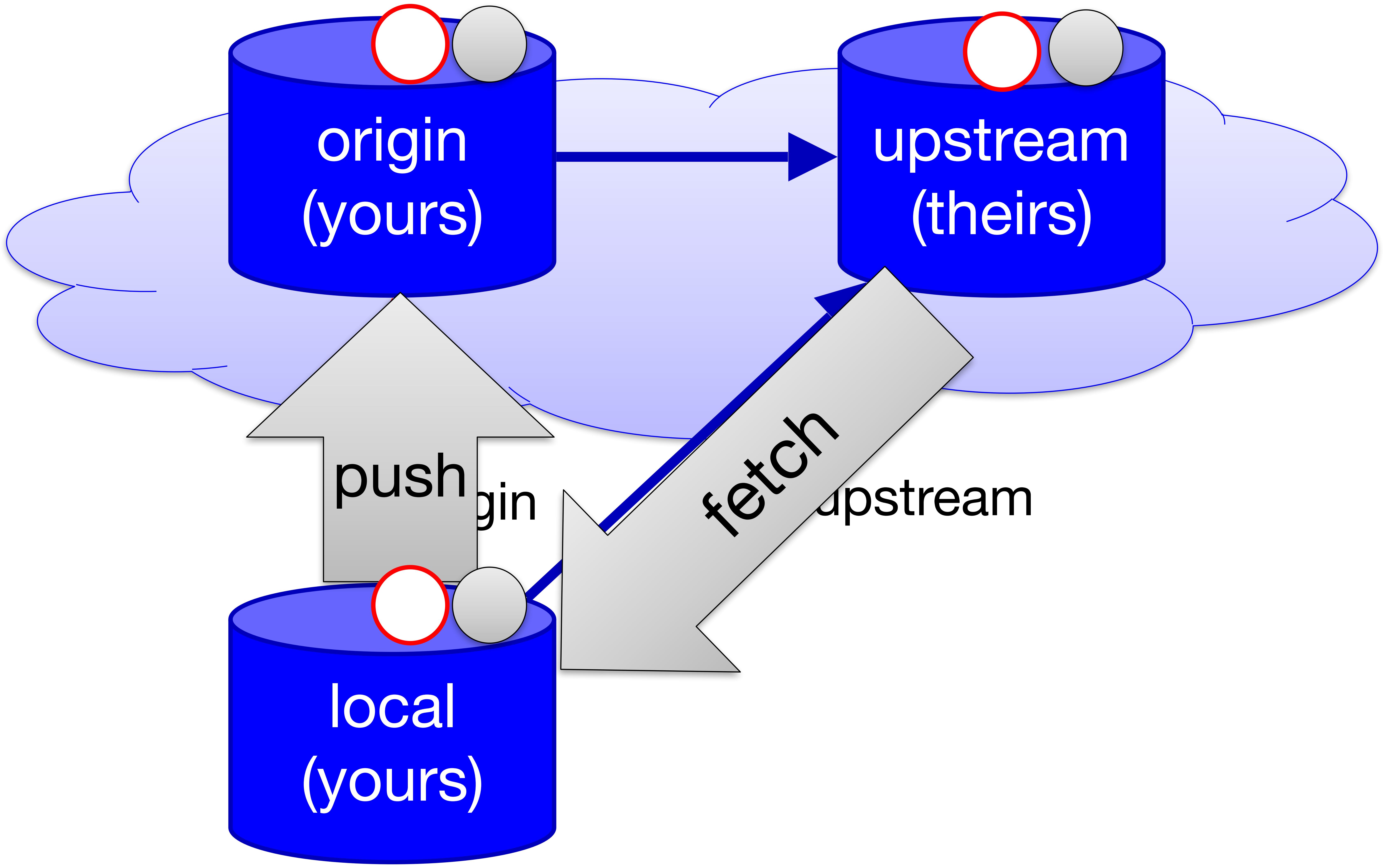
Integrate Changes



Integrate Changes



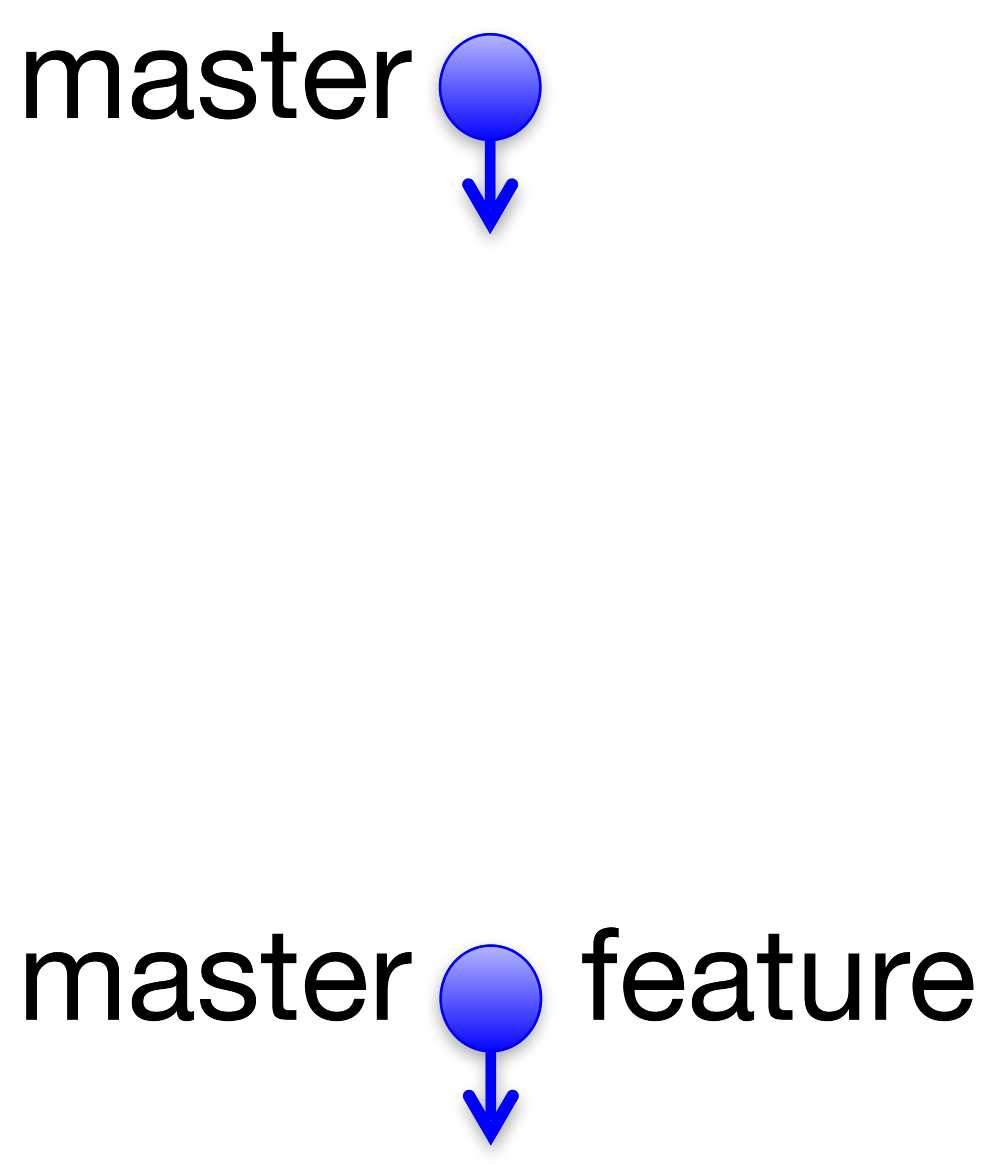
Integrate Changes



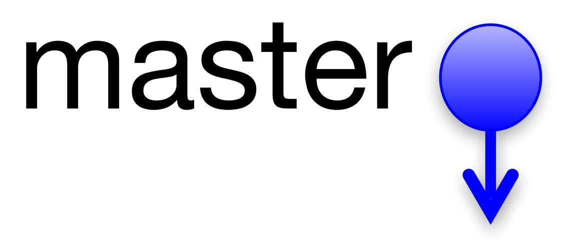
Branches

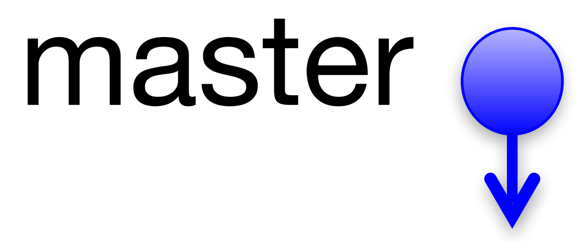
master
master
master

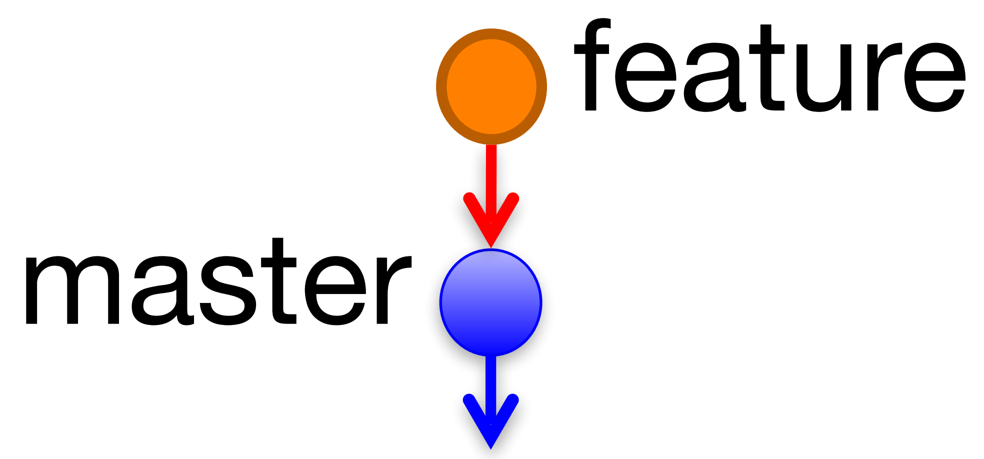
$ git checkout -b feature



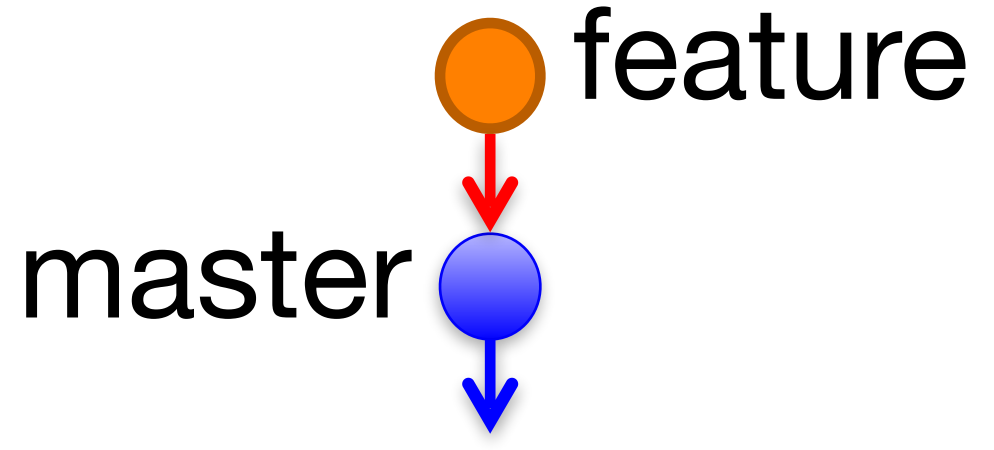
`$ git commit`

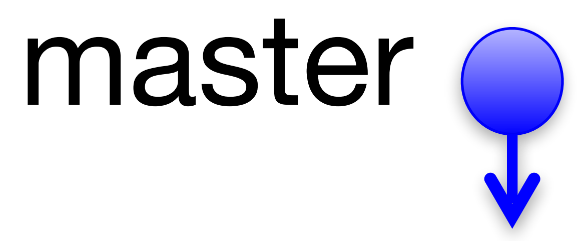


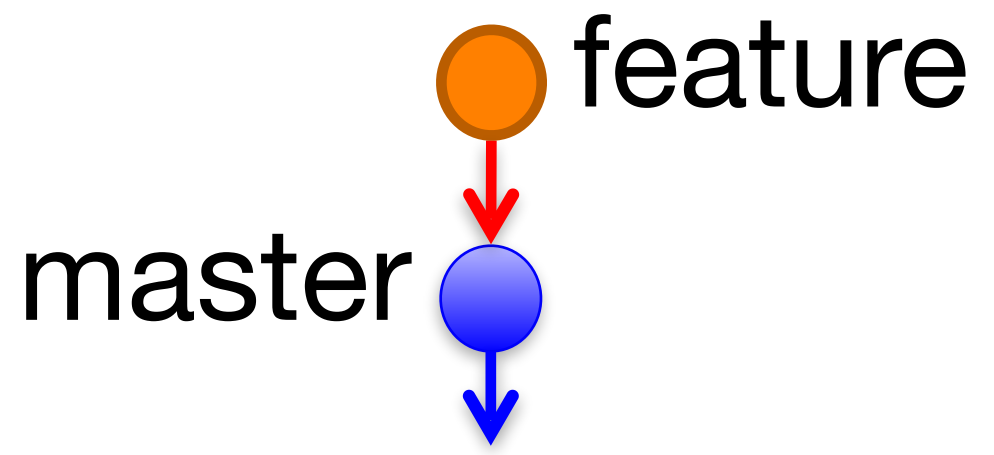




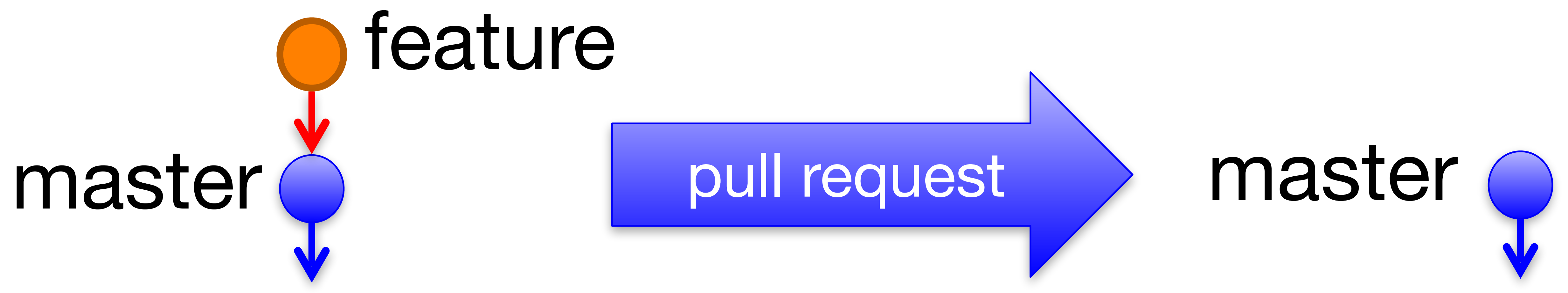
```
$ git push -u origin feature
```

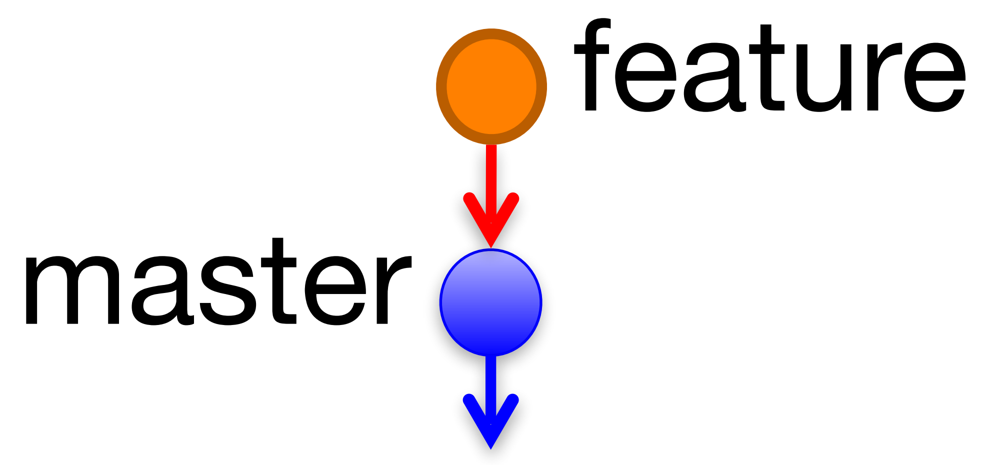




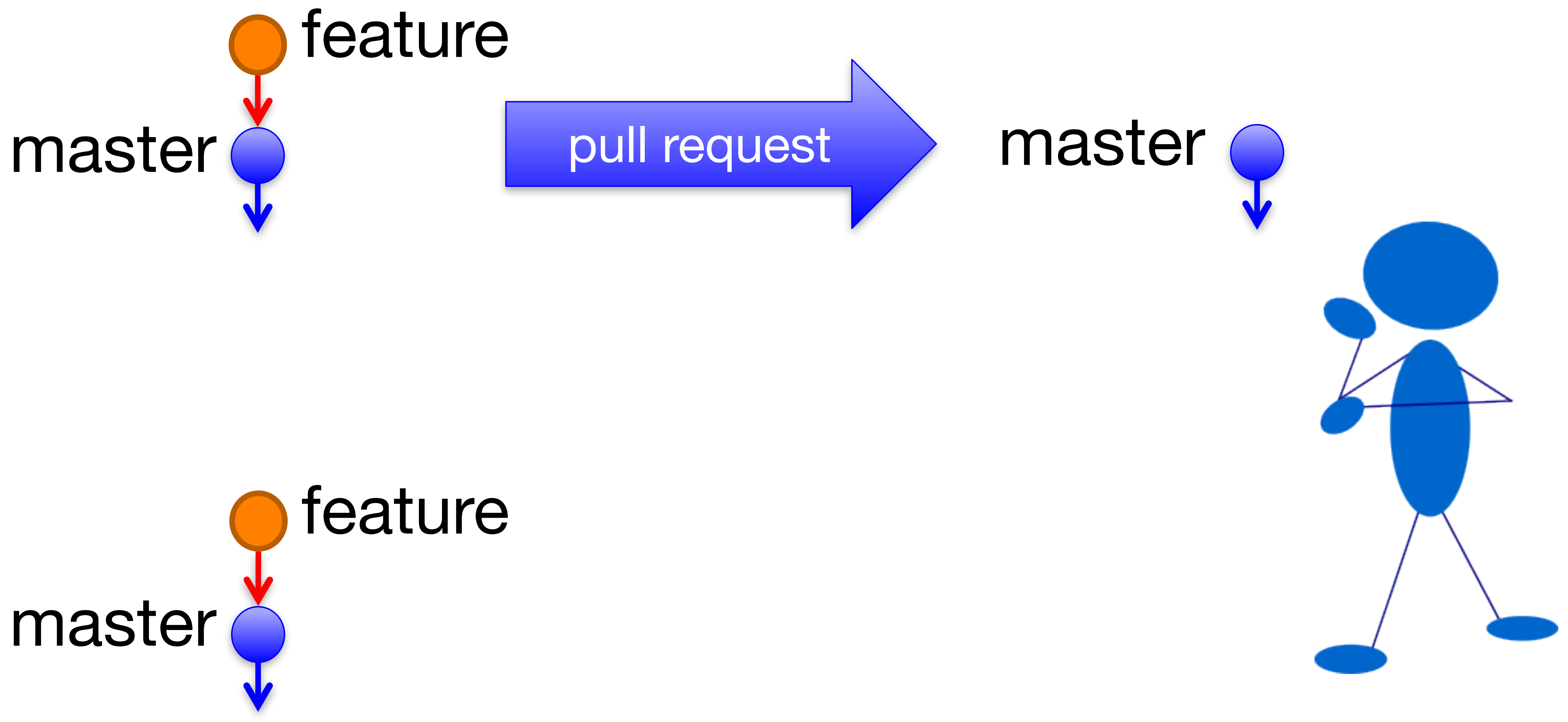


New pull request

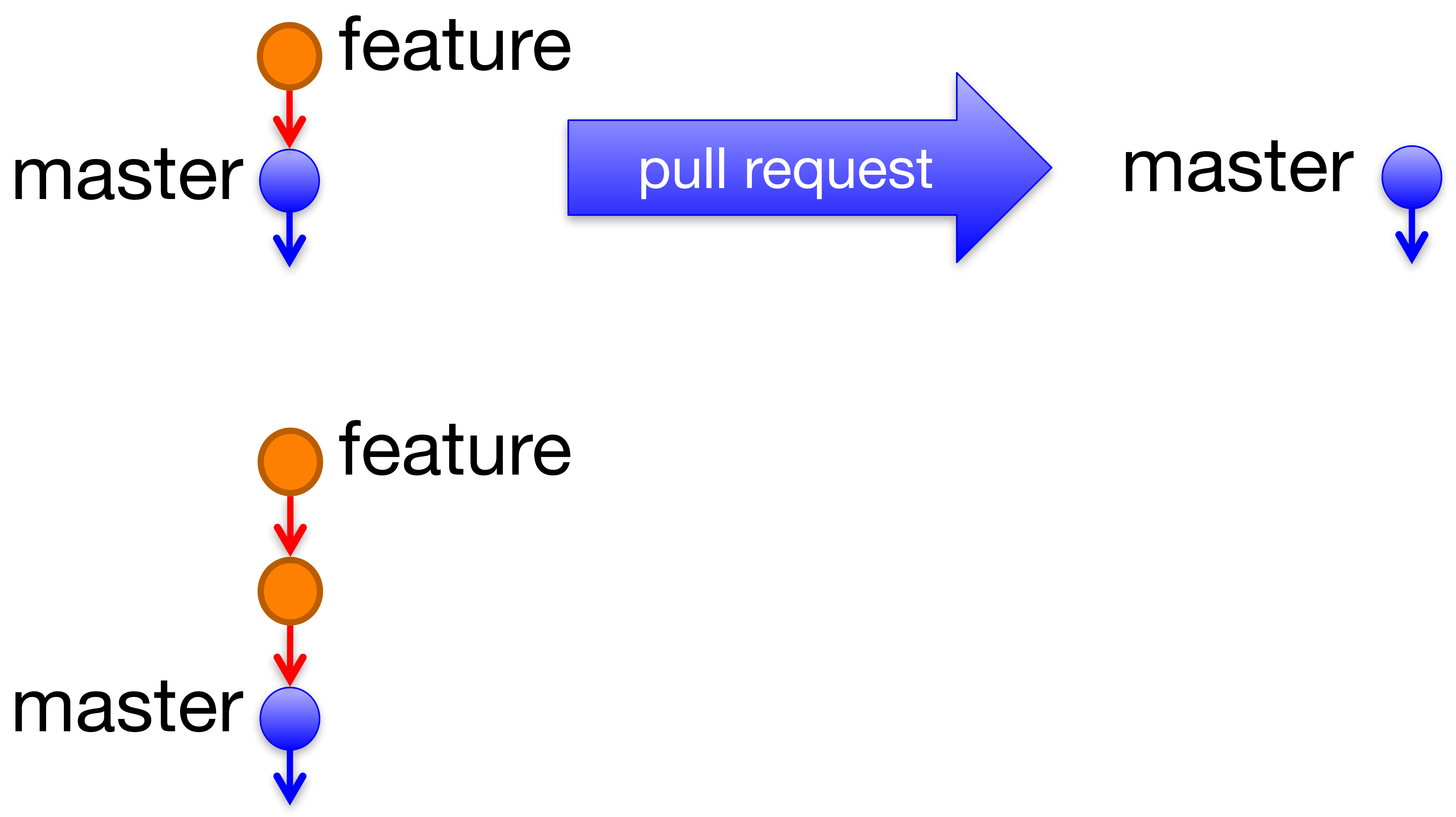

feature
master
pull request
master


feature
master

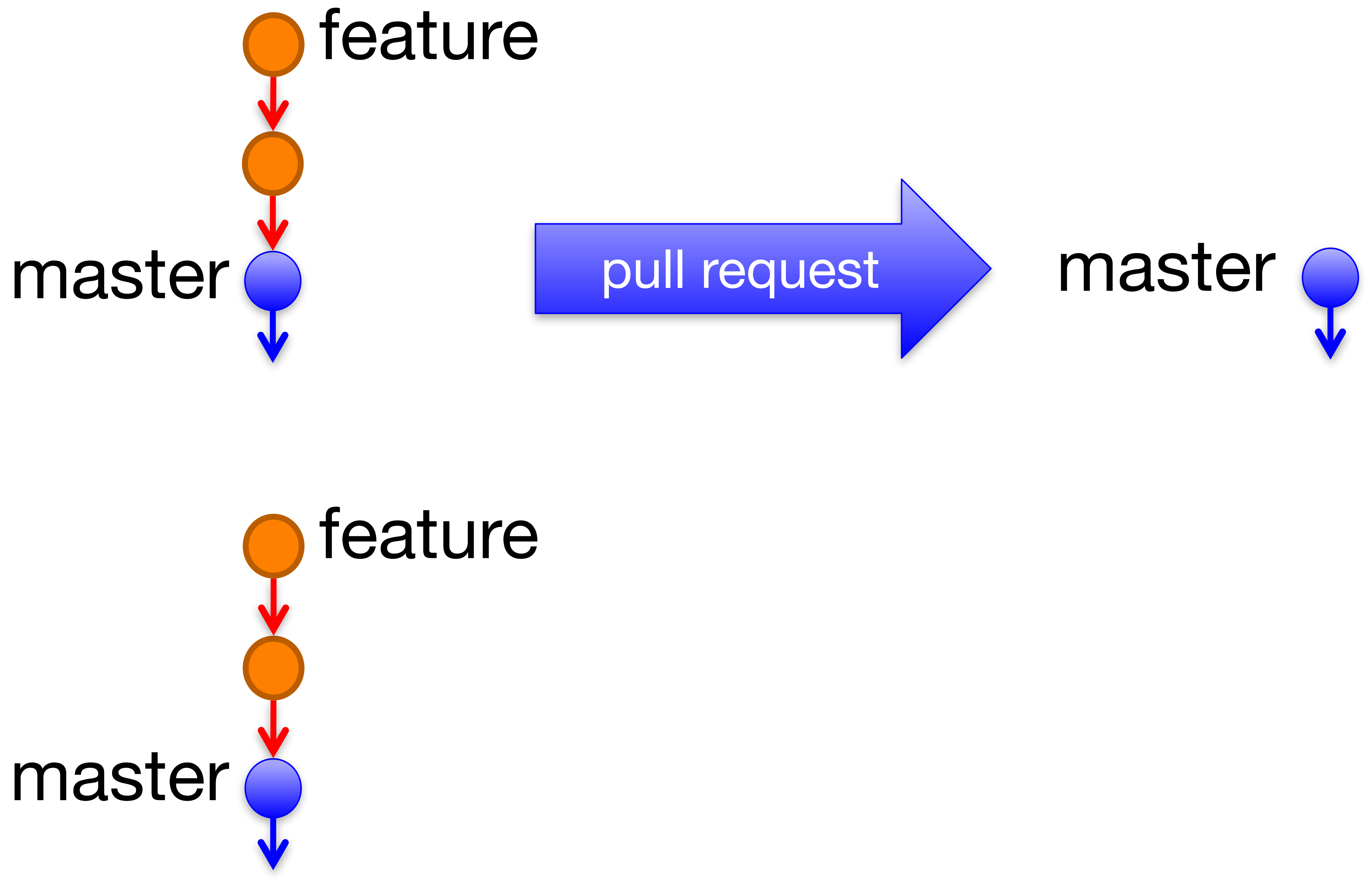
Great idea, now can you do it more like this?



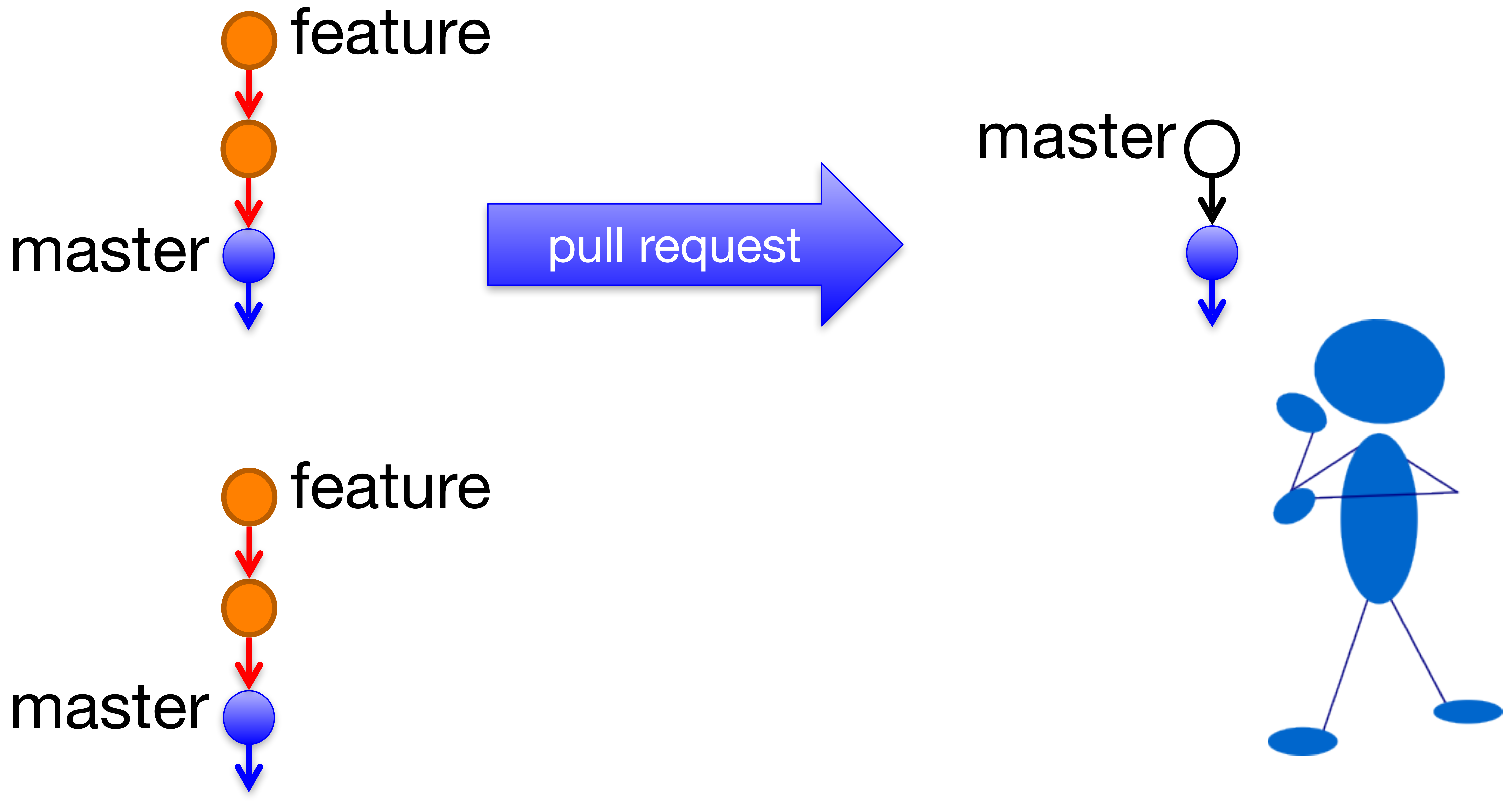
$ git commit



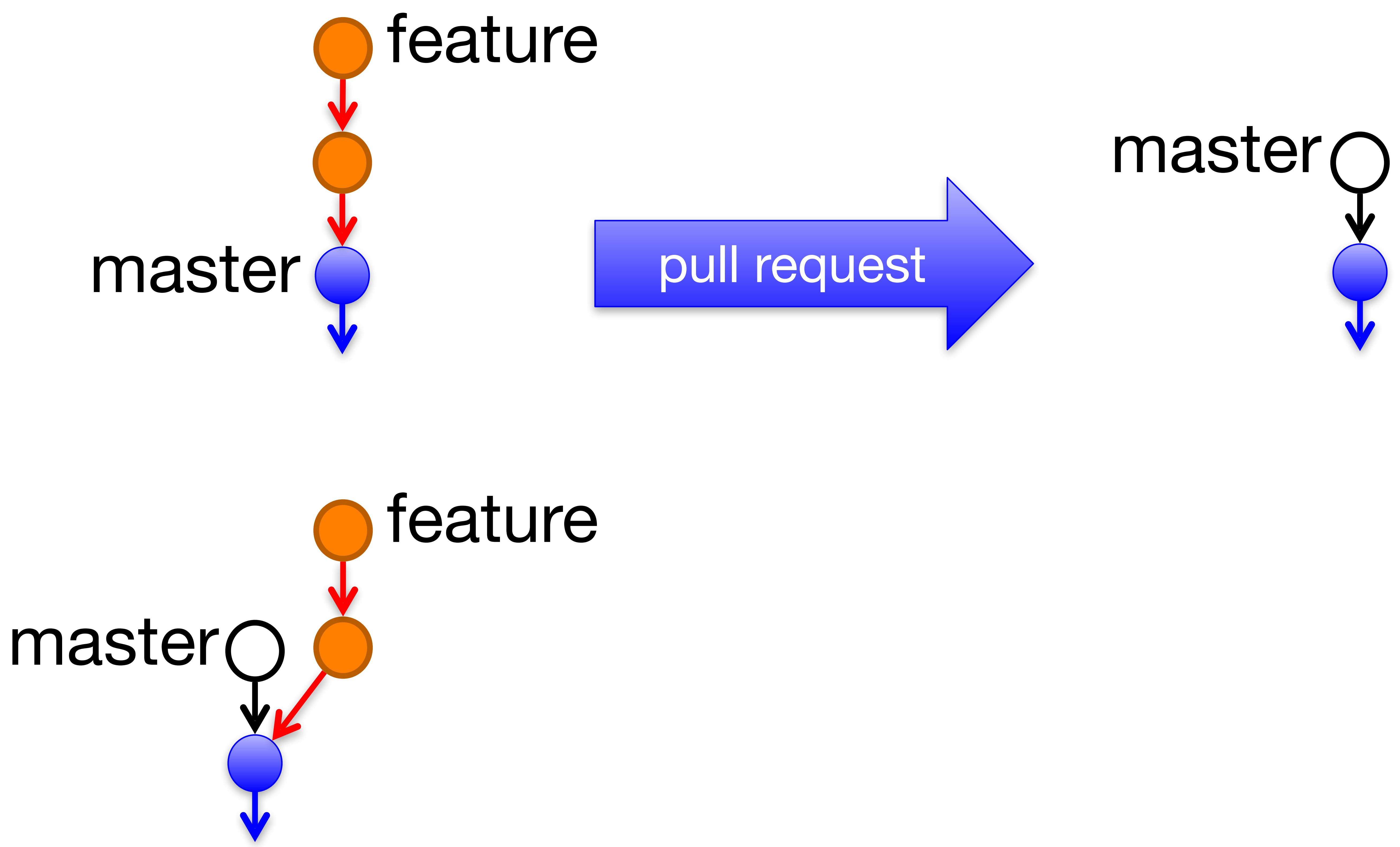
`$ git push origin feature`



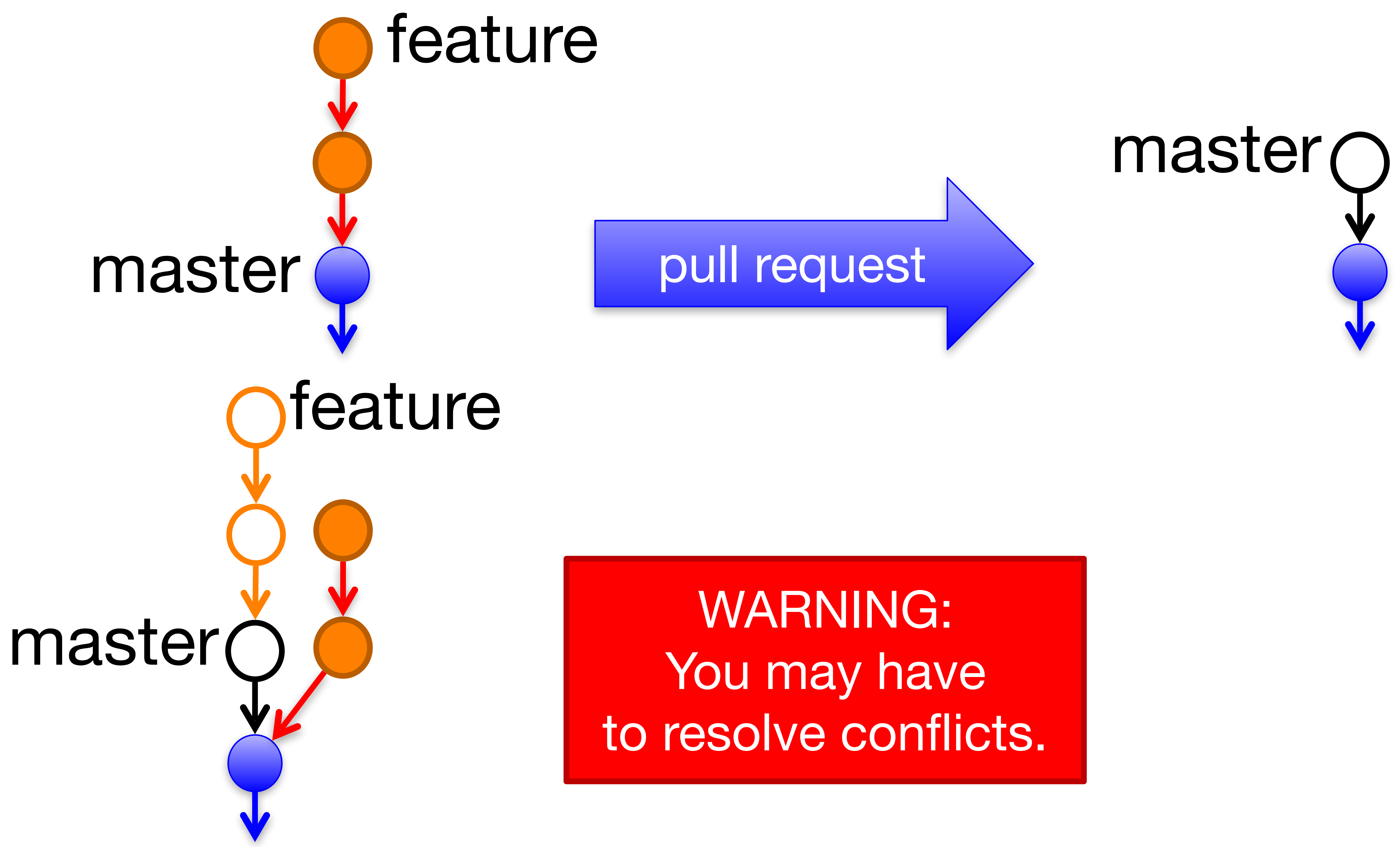
Awesome, but please update with new changes in master



`$ git fetch upstream master:master`

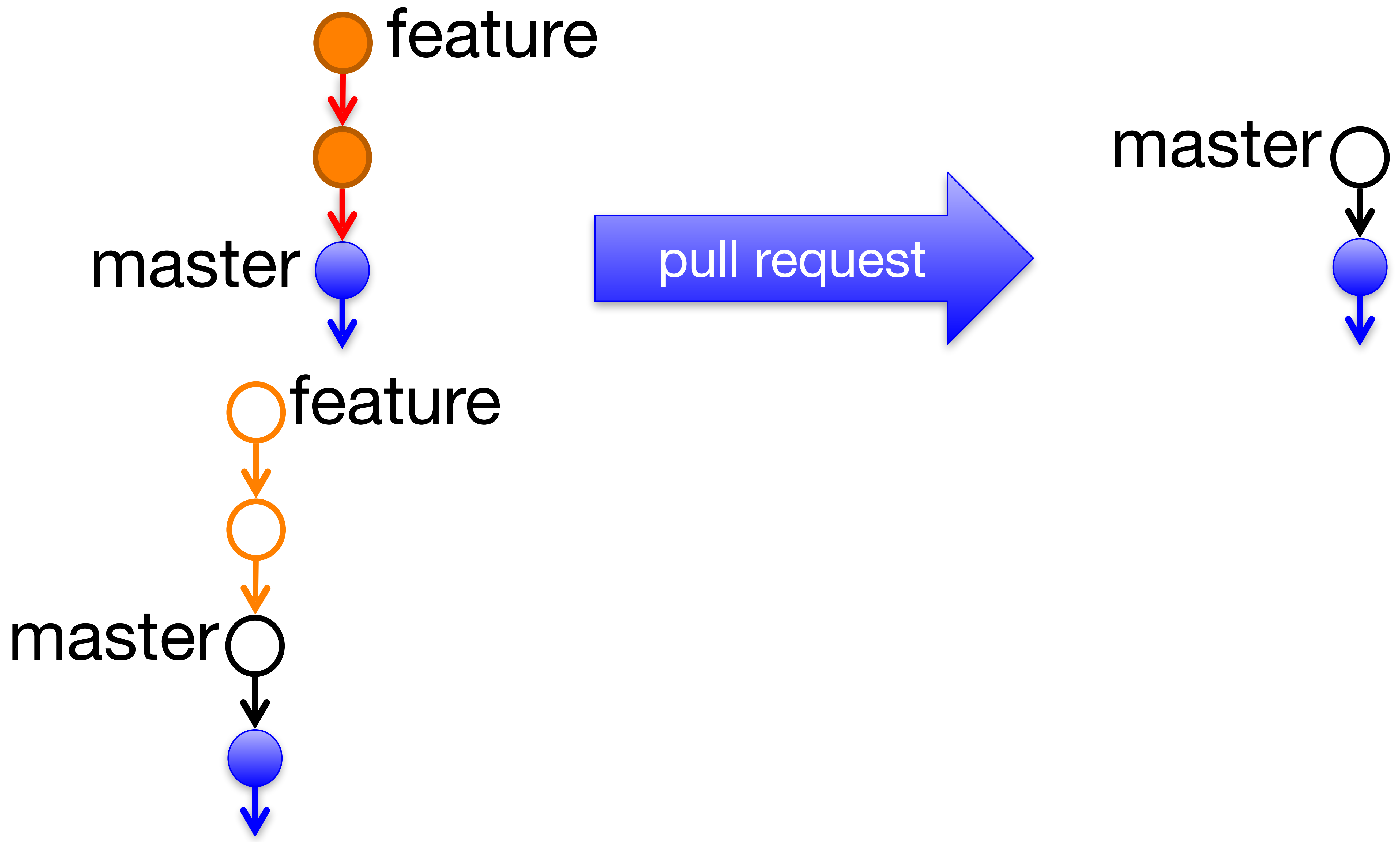


`$ git rebase master`

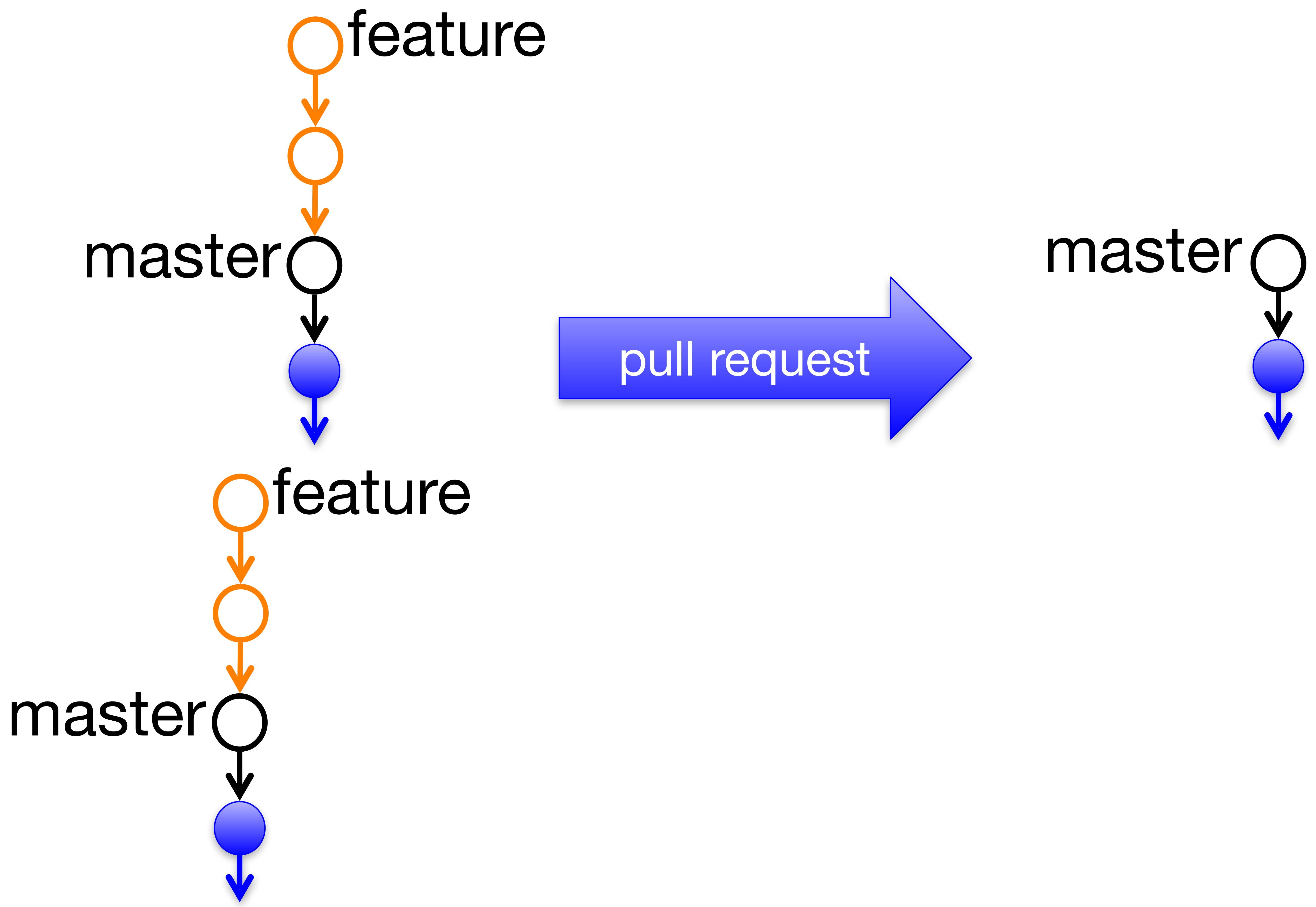


WARNING:
You may have
to resolve conflicts.

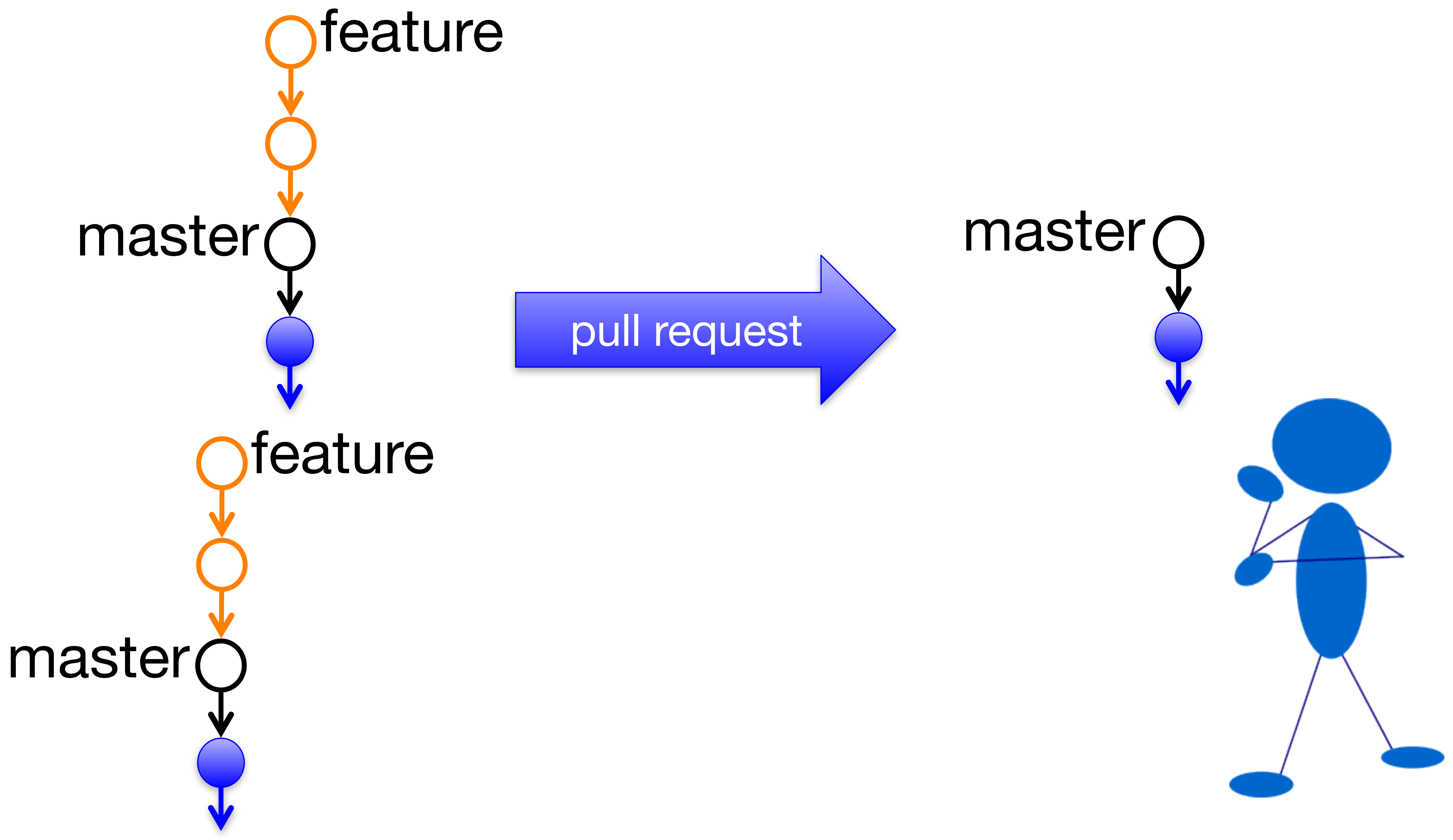
`$ git rebase master`



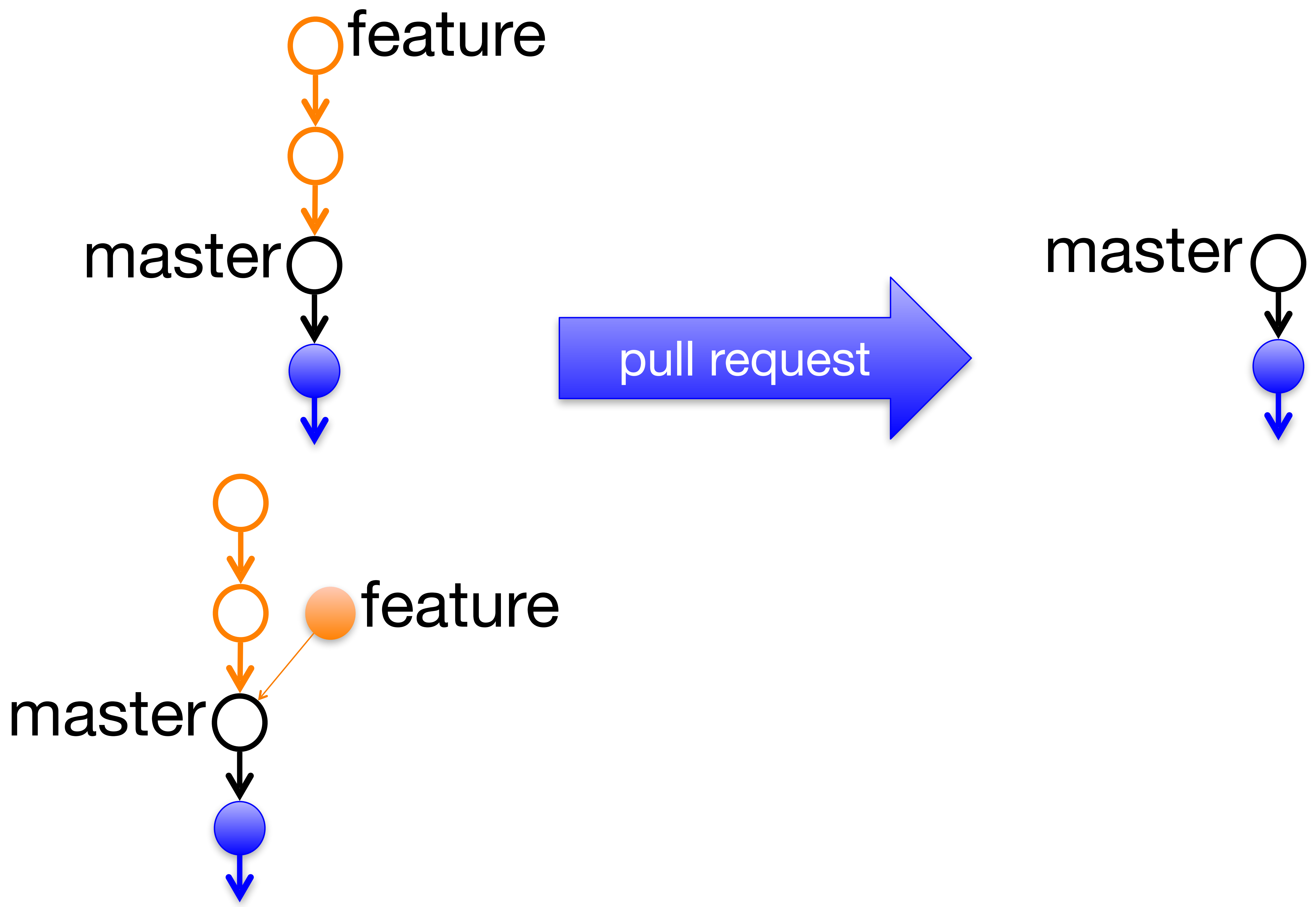
```
$ git push -f origin master feature
```



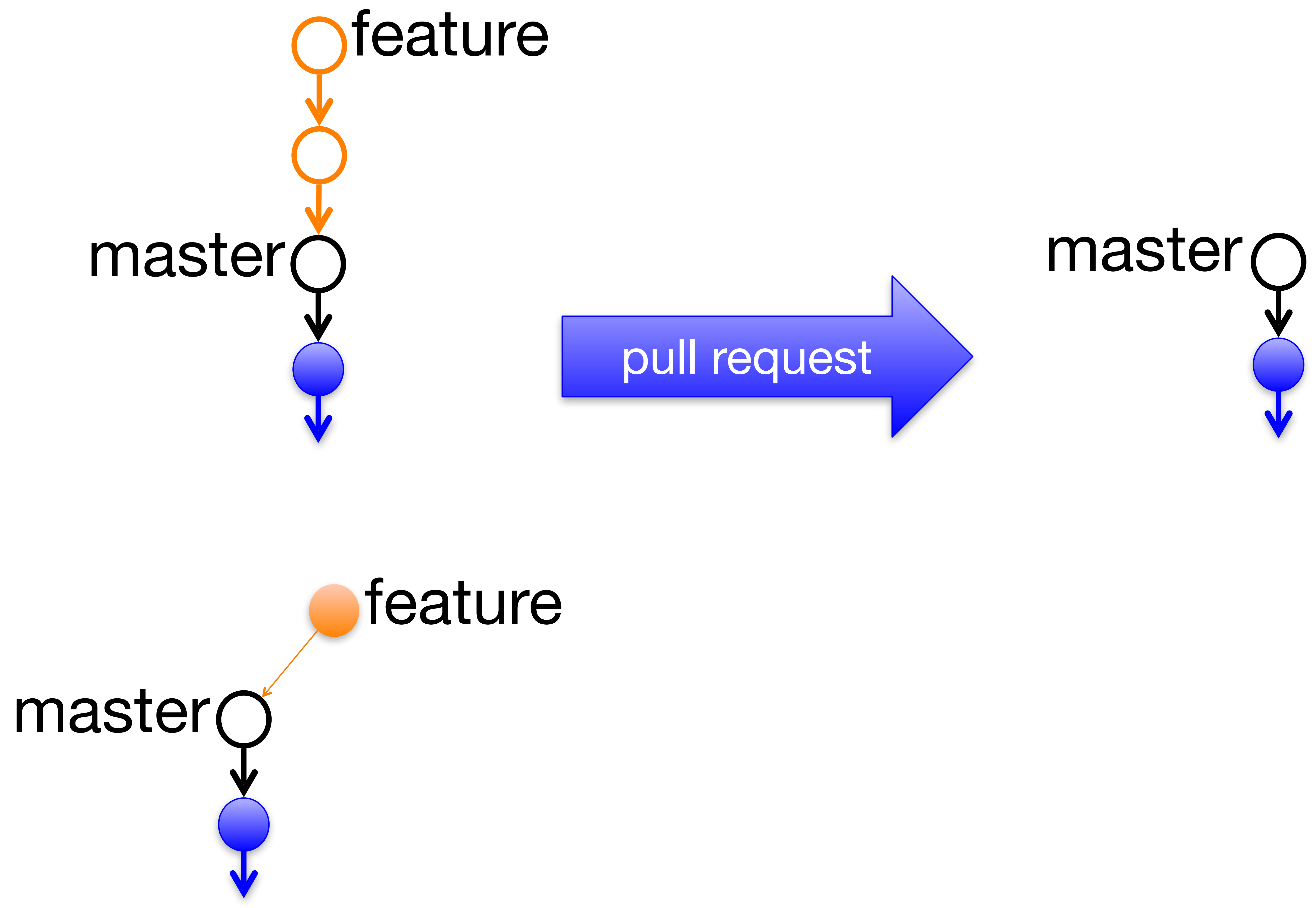
Great. Please squash your commits.



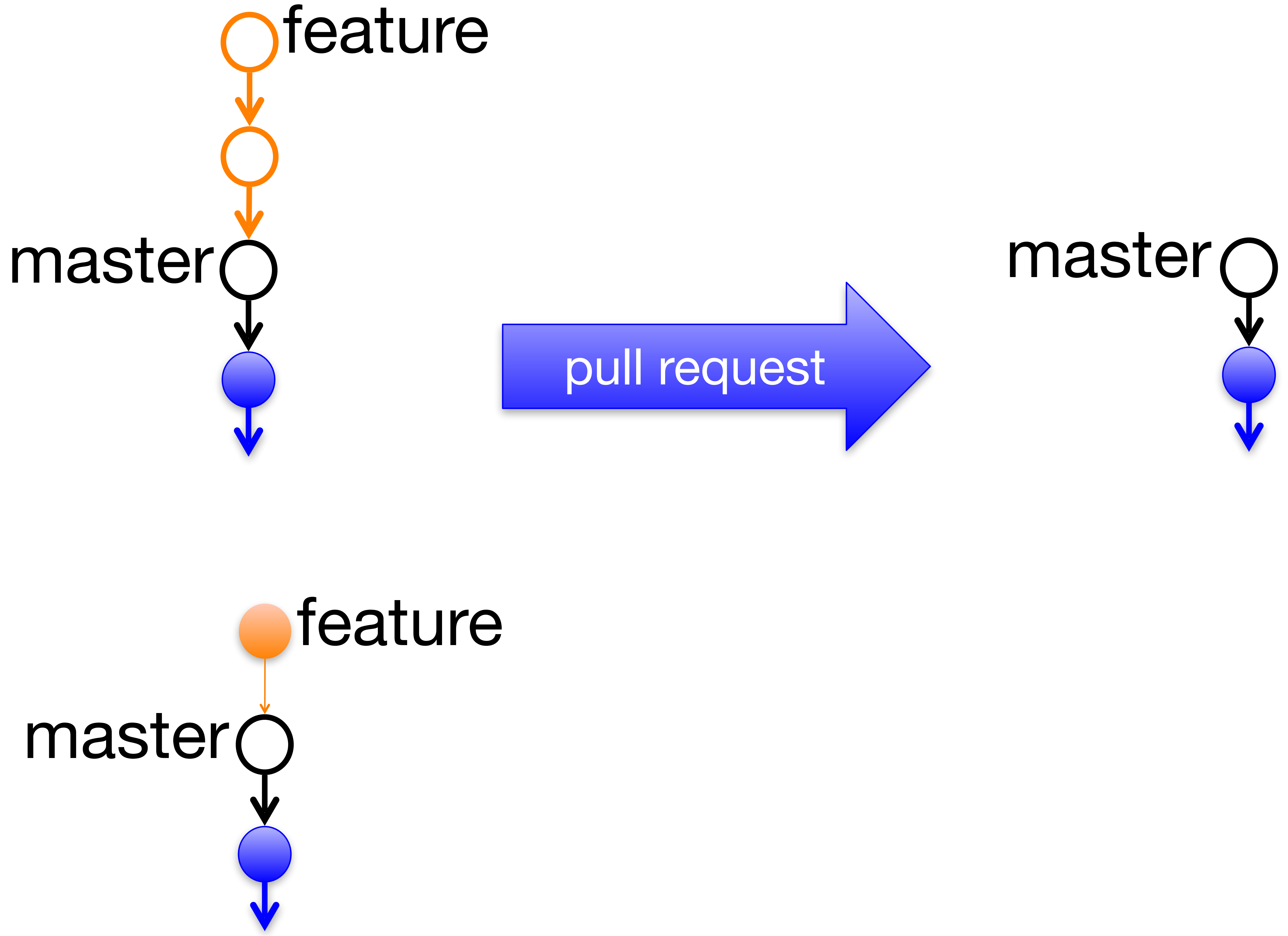
$ git rebase –i master



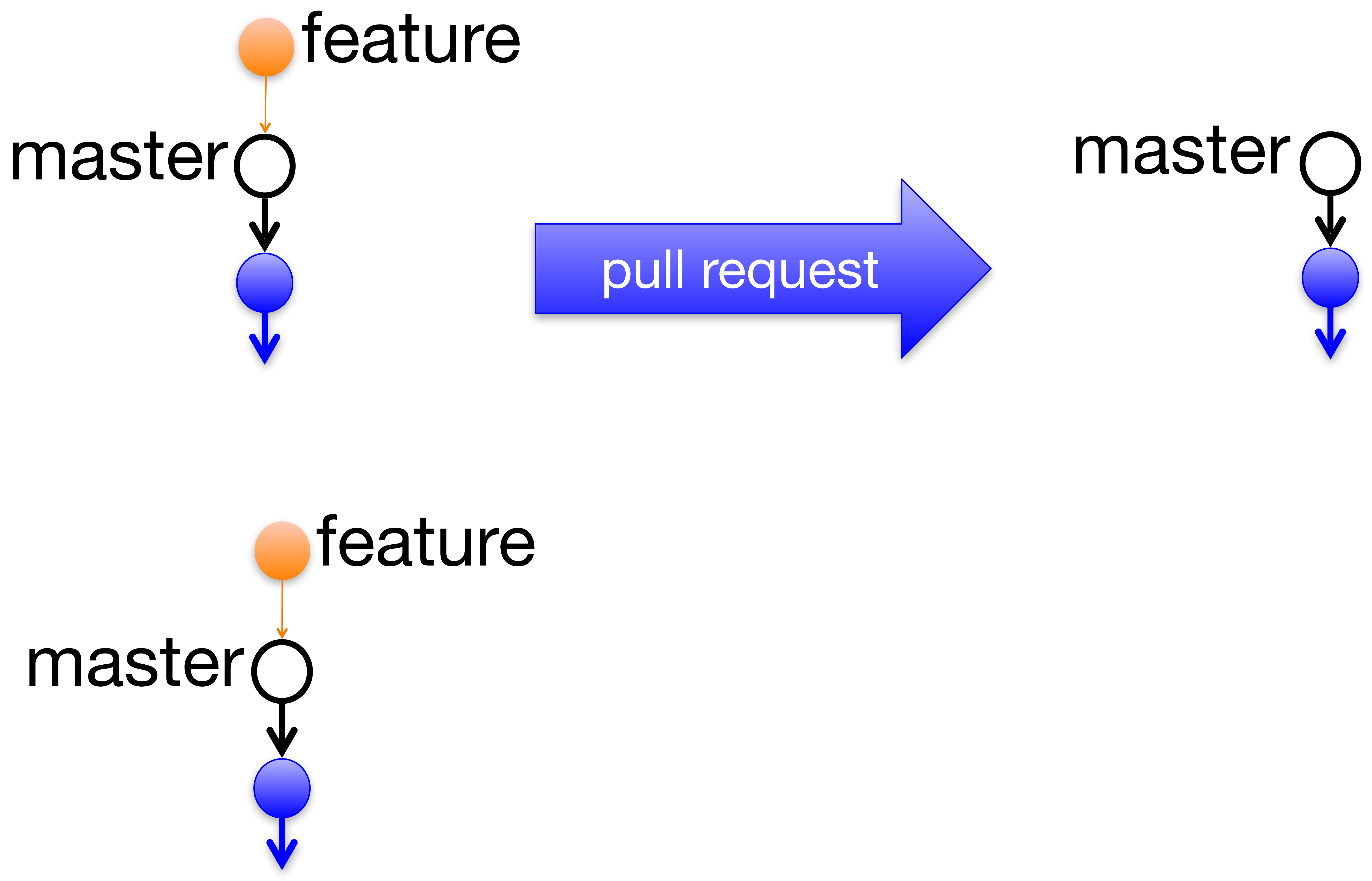
$ git rebase –i master



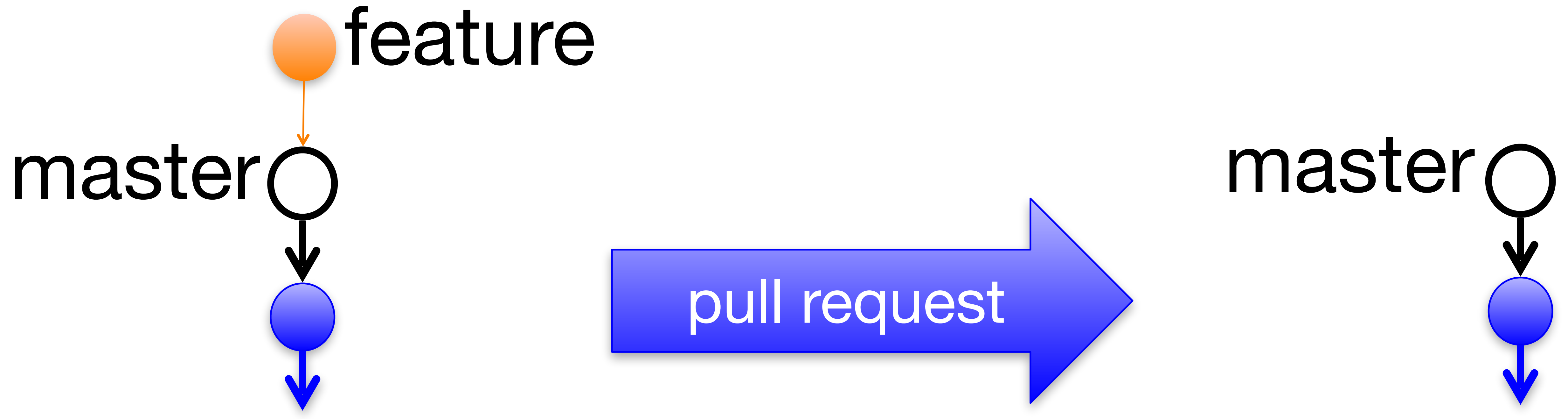
`$ git rebase –i master`

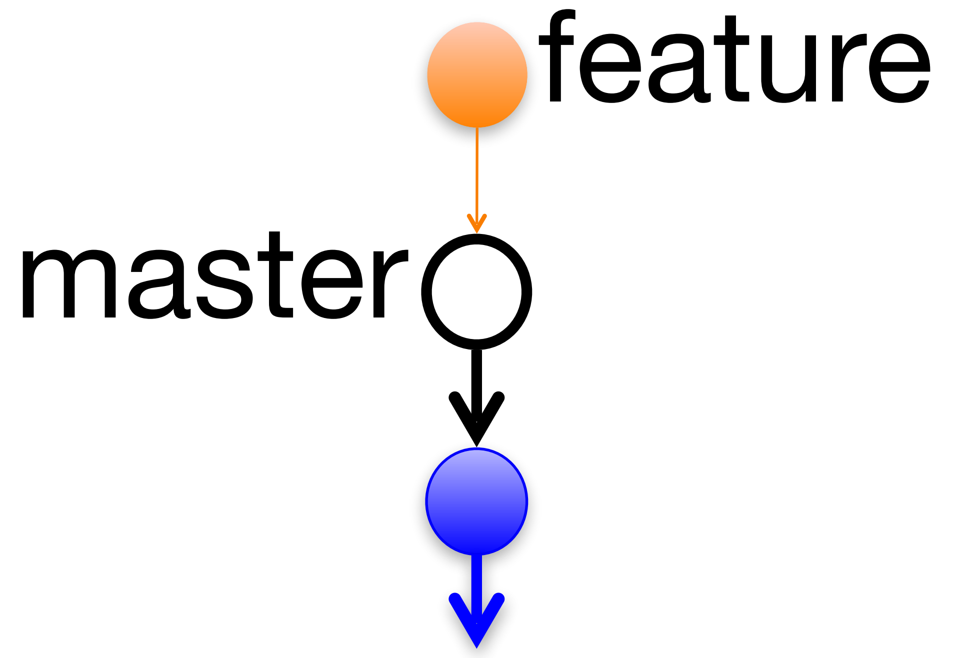


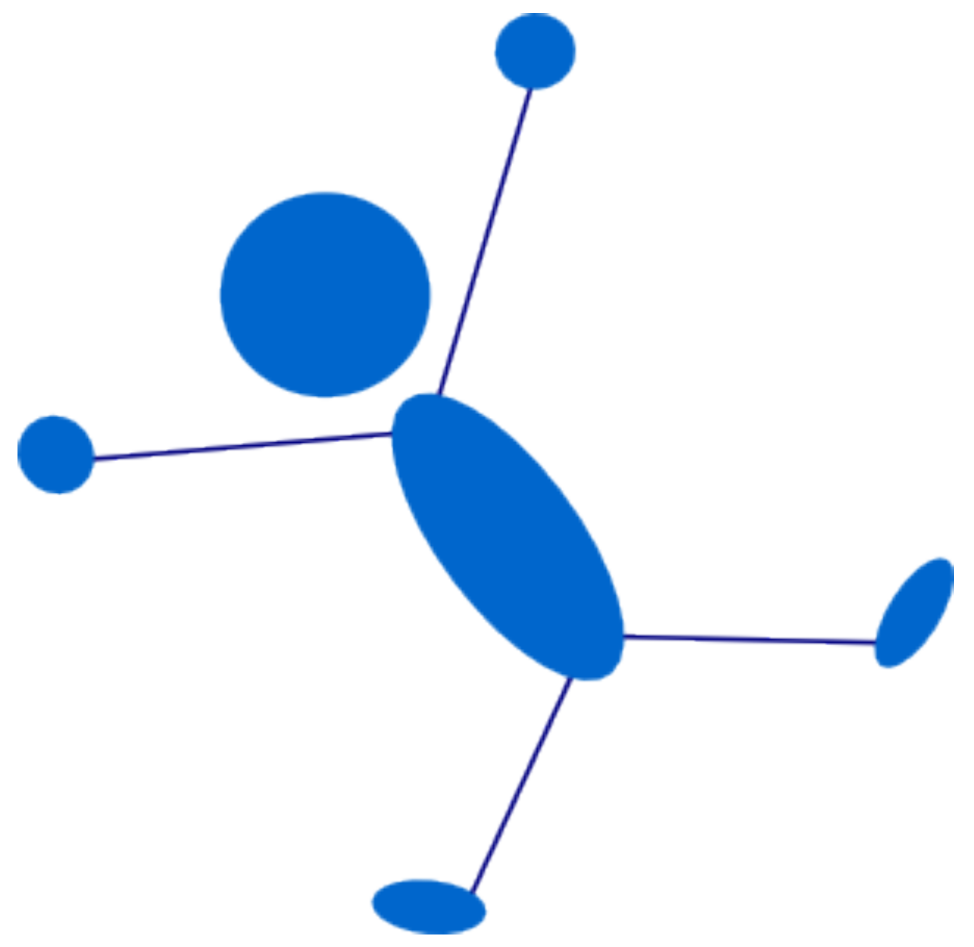
```
$ git push -f origin feature
```



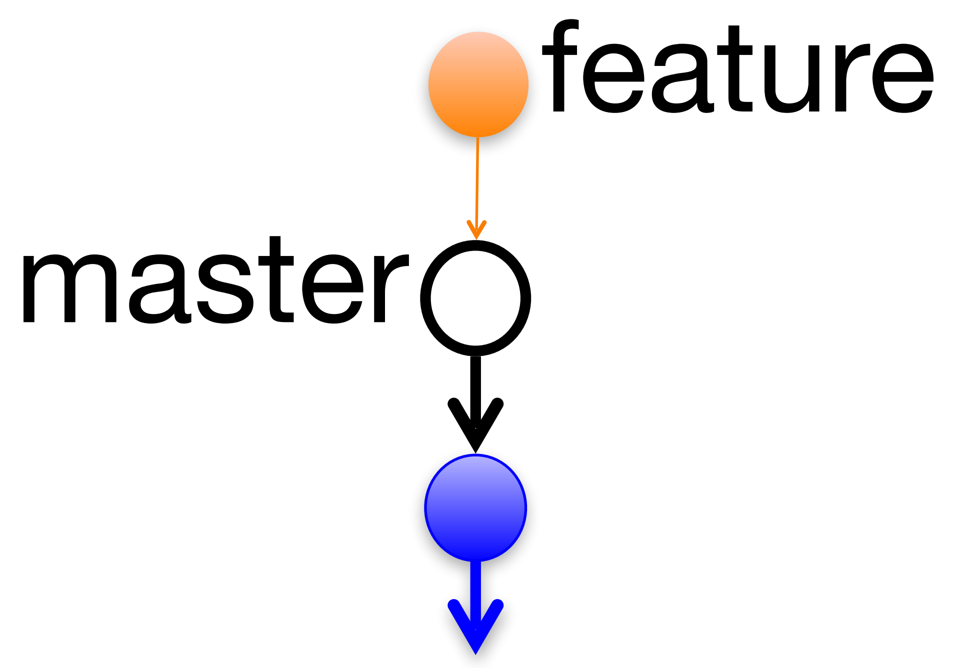
Perfect, I accept!

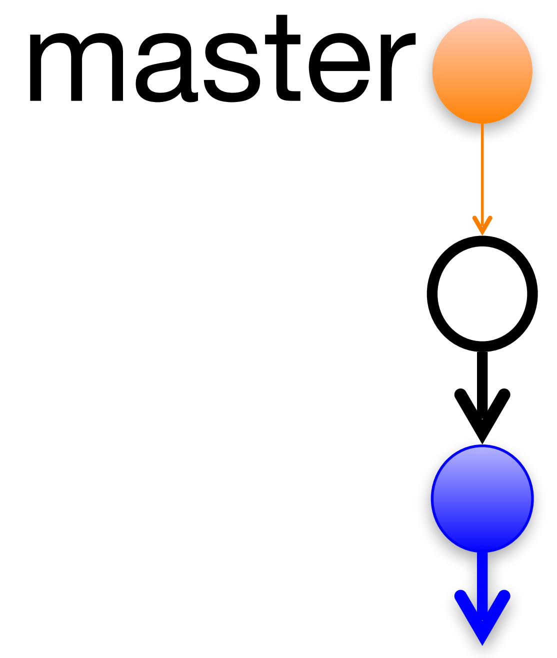


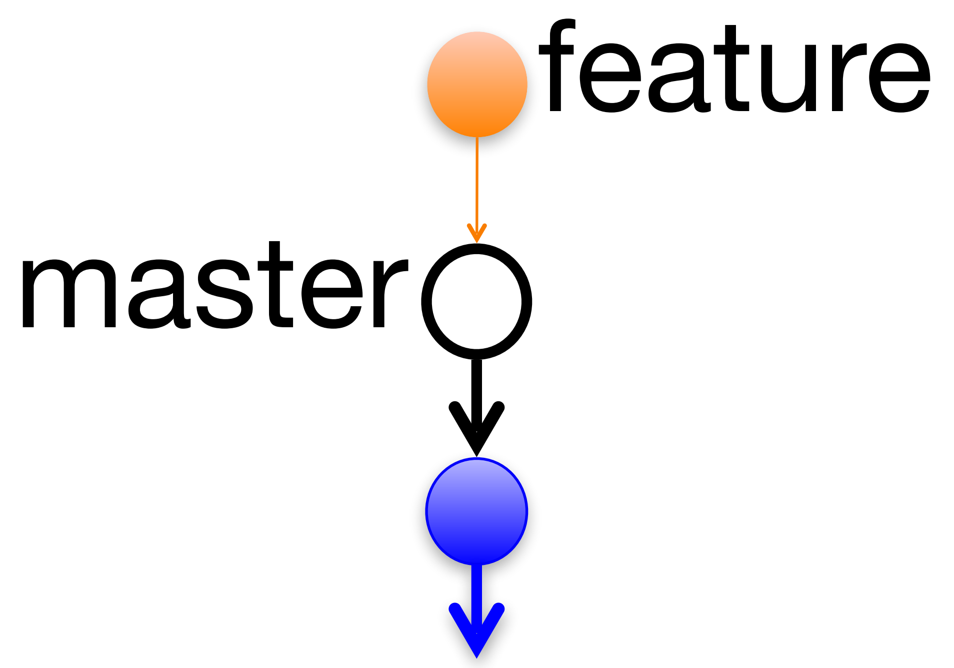




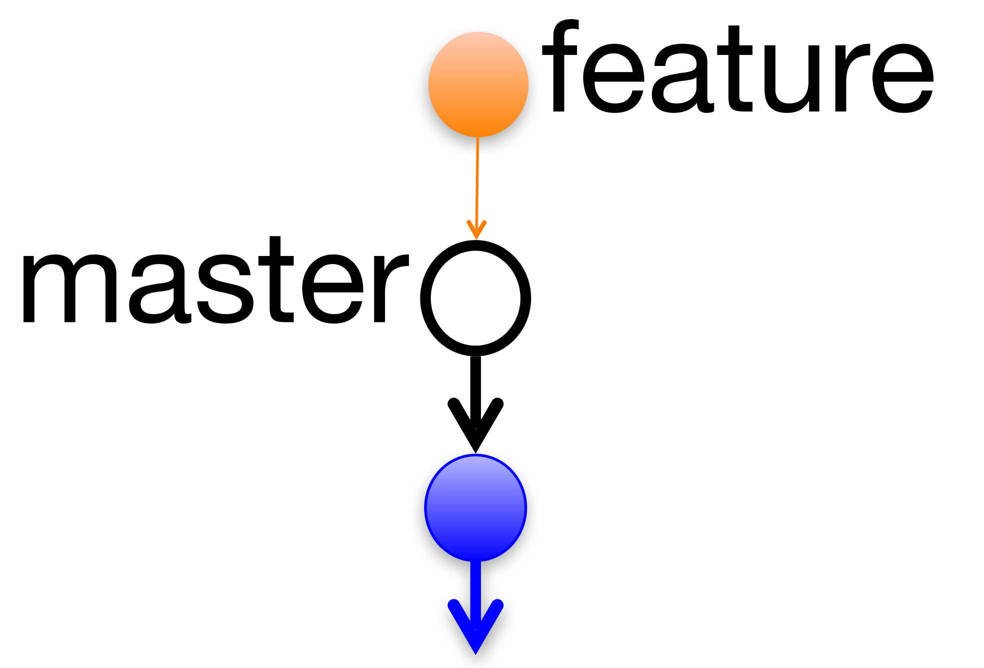
Time to Clean Up

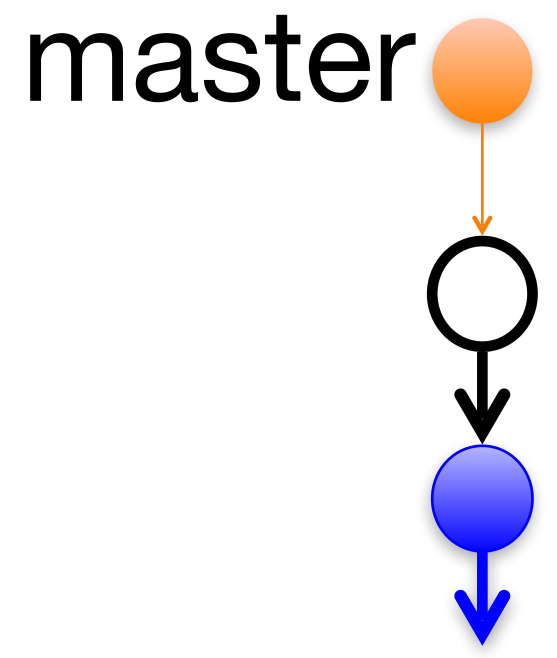


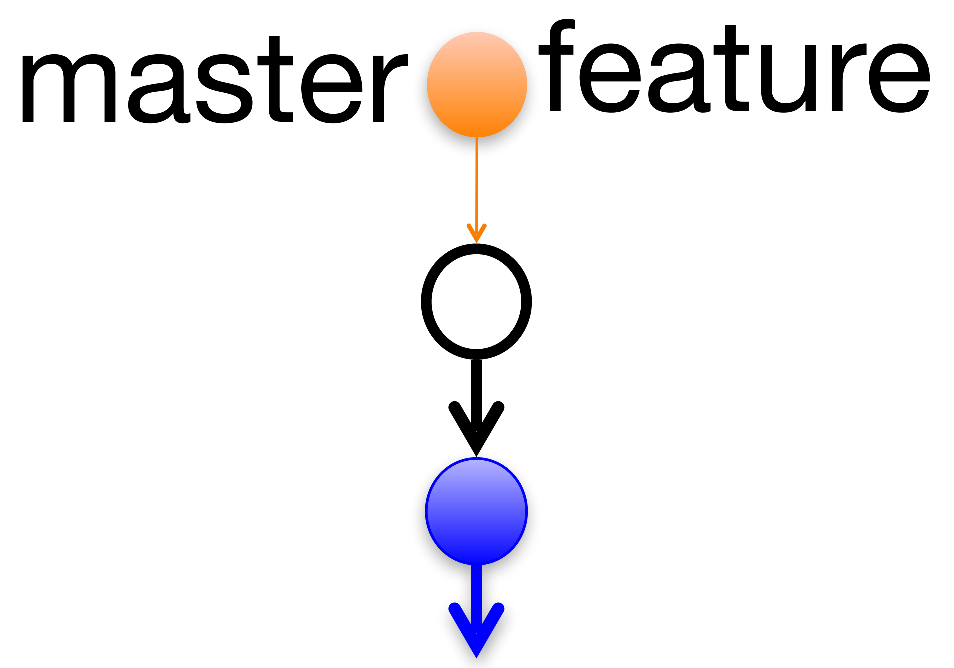




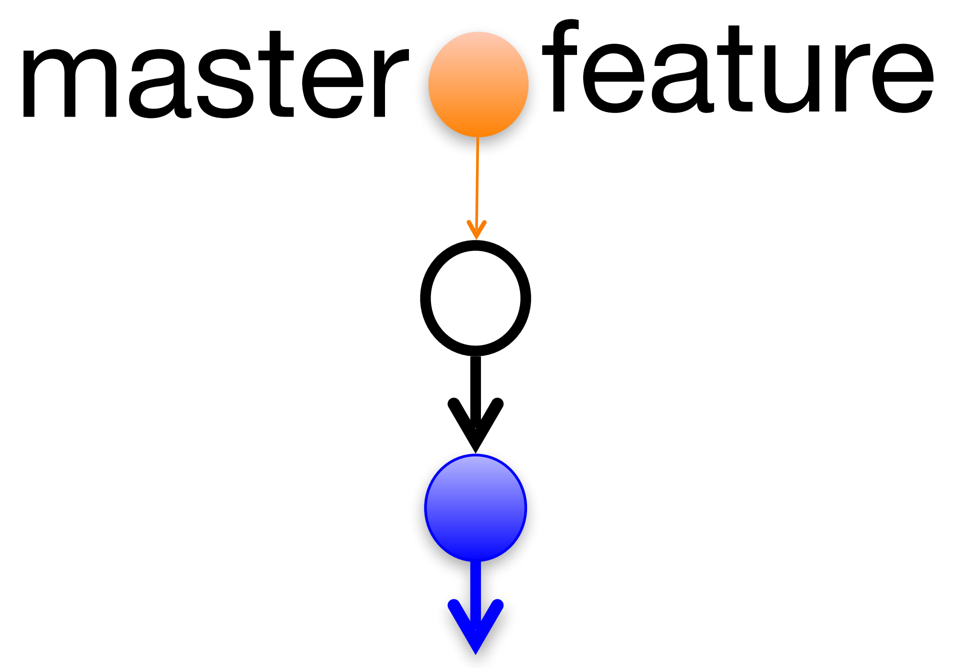
```
$ git fetch upstream master:master
```

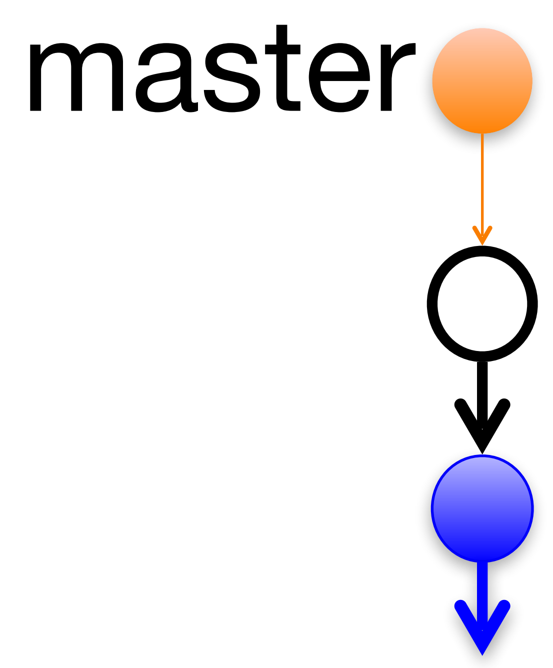


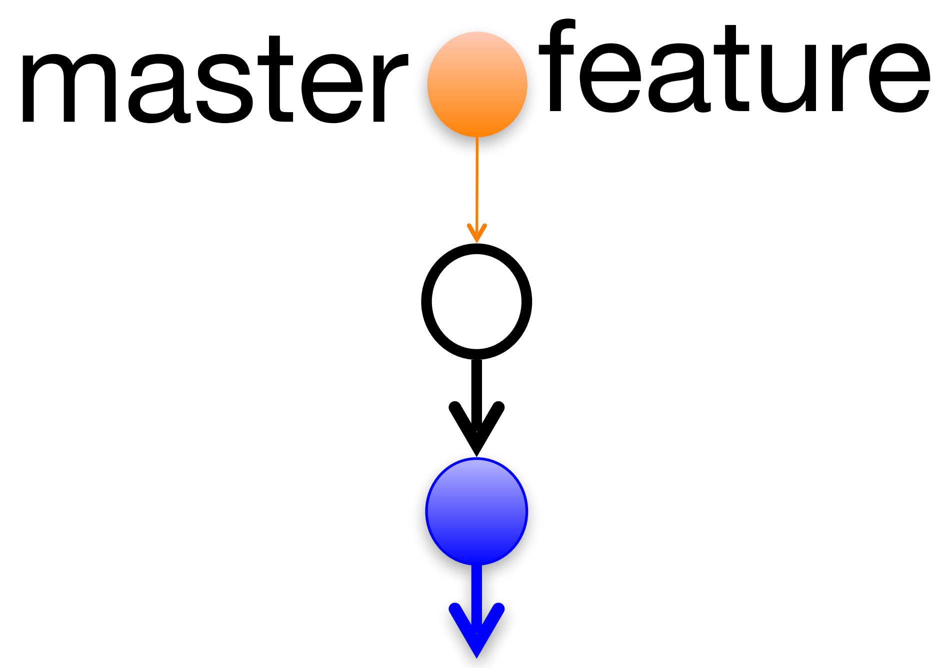




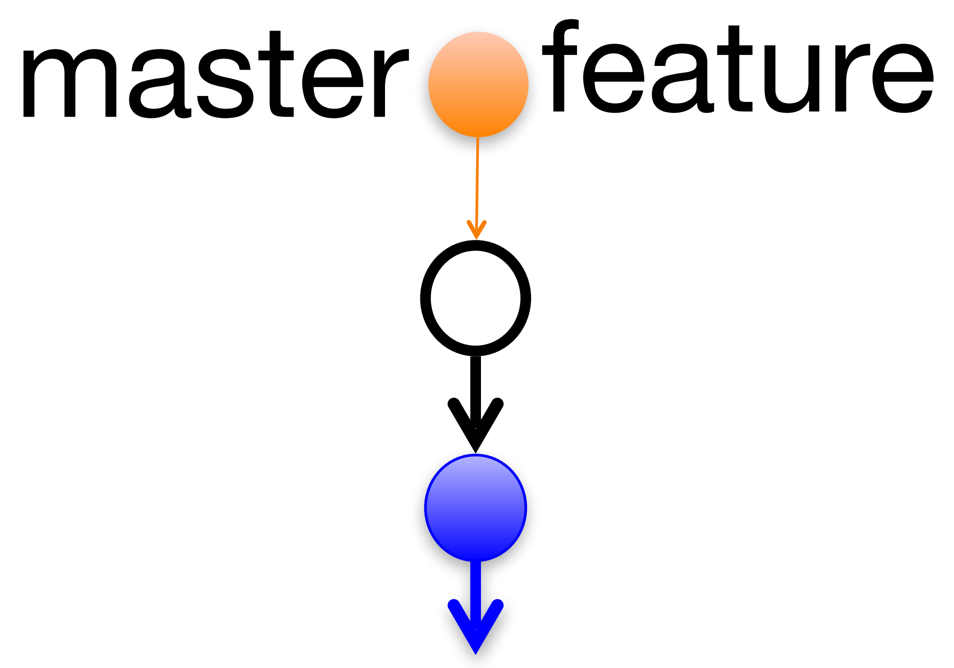
`$ git push origin master`

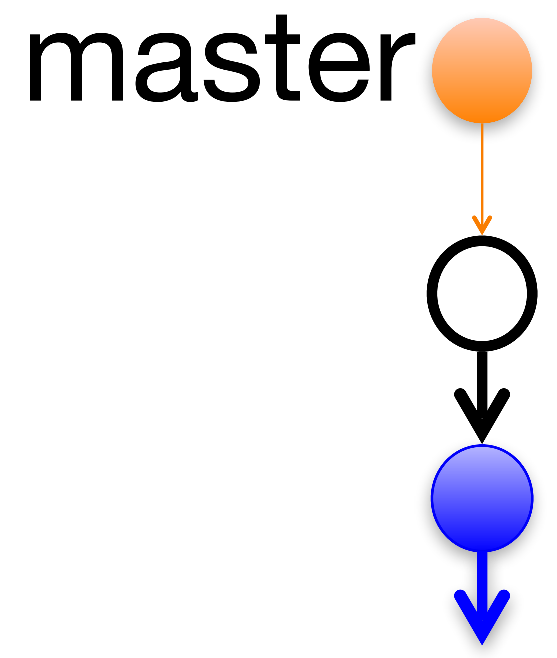


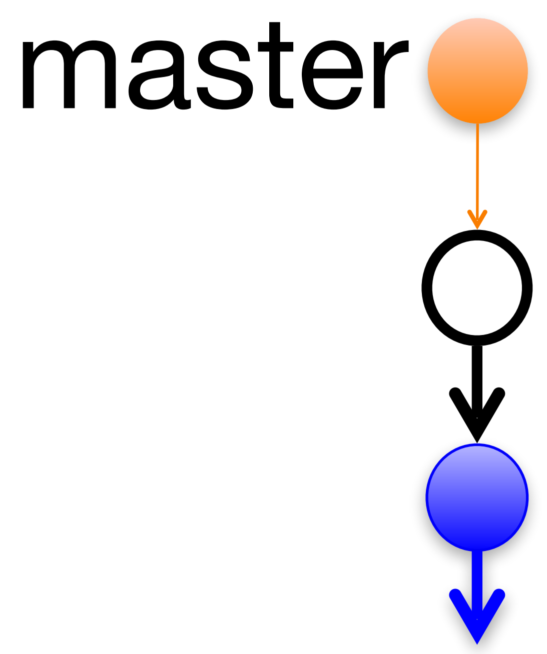




```
$ git checkout master
$ git branch -d feature
```







`$ git push origin -d feature`