UNIVERSITÄT
BERN

Distributed version control with git — a brief introduction

Andrei Chis

based on slides by
Oscar Nierstrasz

What is git?

Distributed version control system

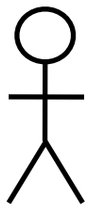
What does a version control system do?

Tracks the history of a collection of files

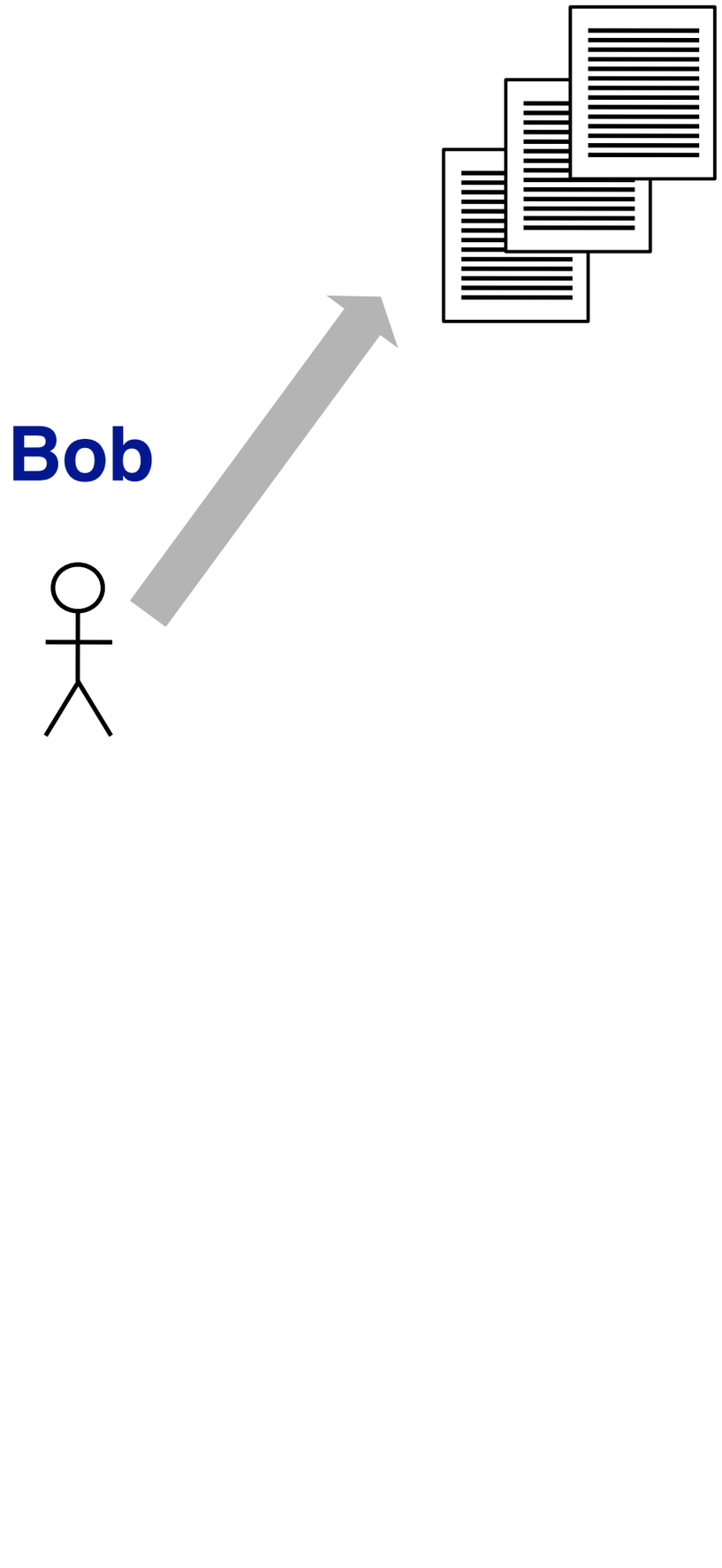
Can revert the collection of files to another version

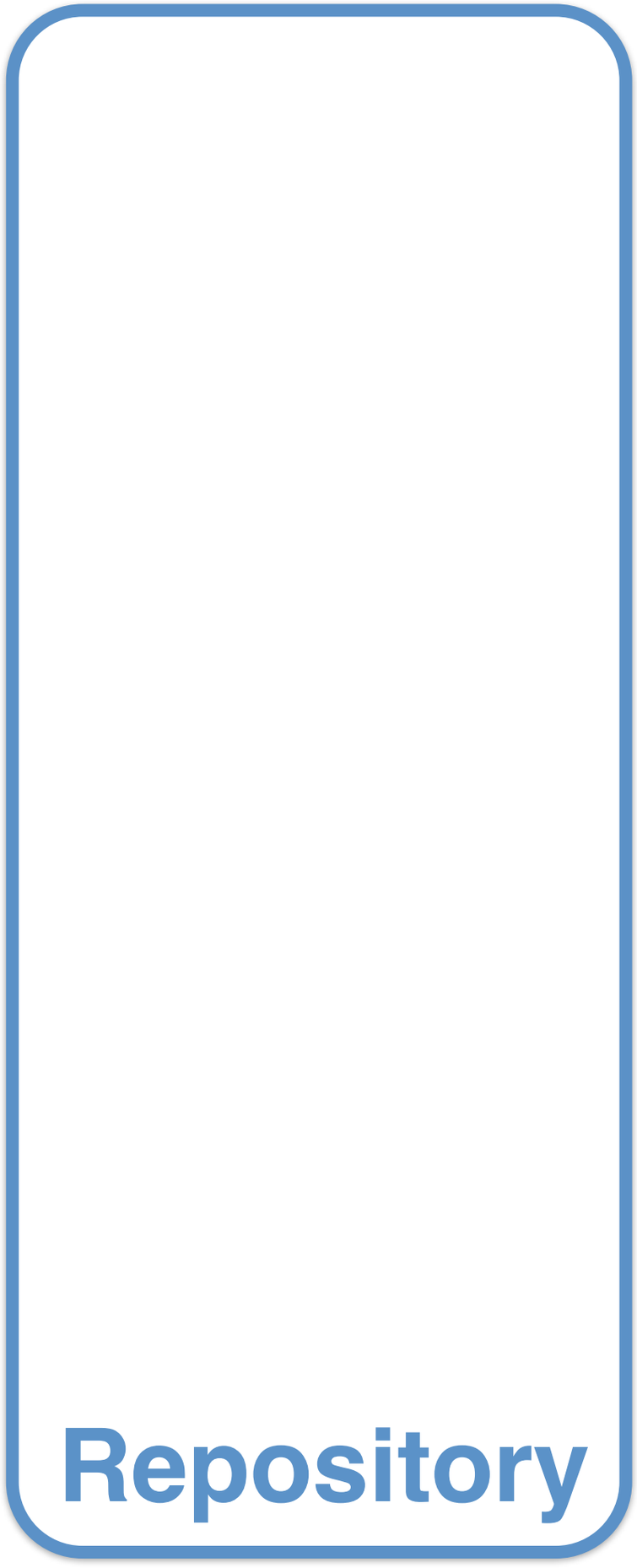
Allows a team of people to work together on the same files

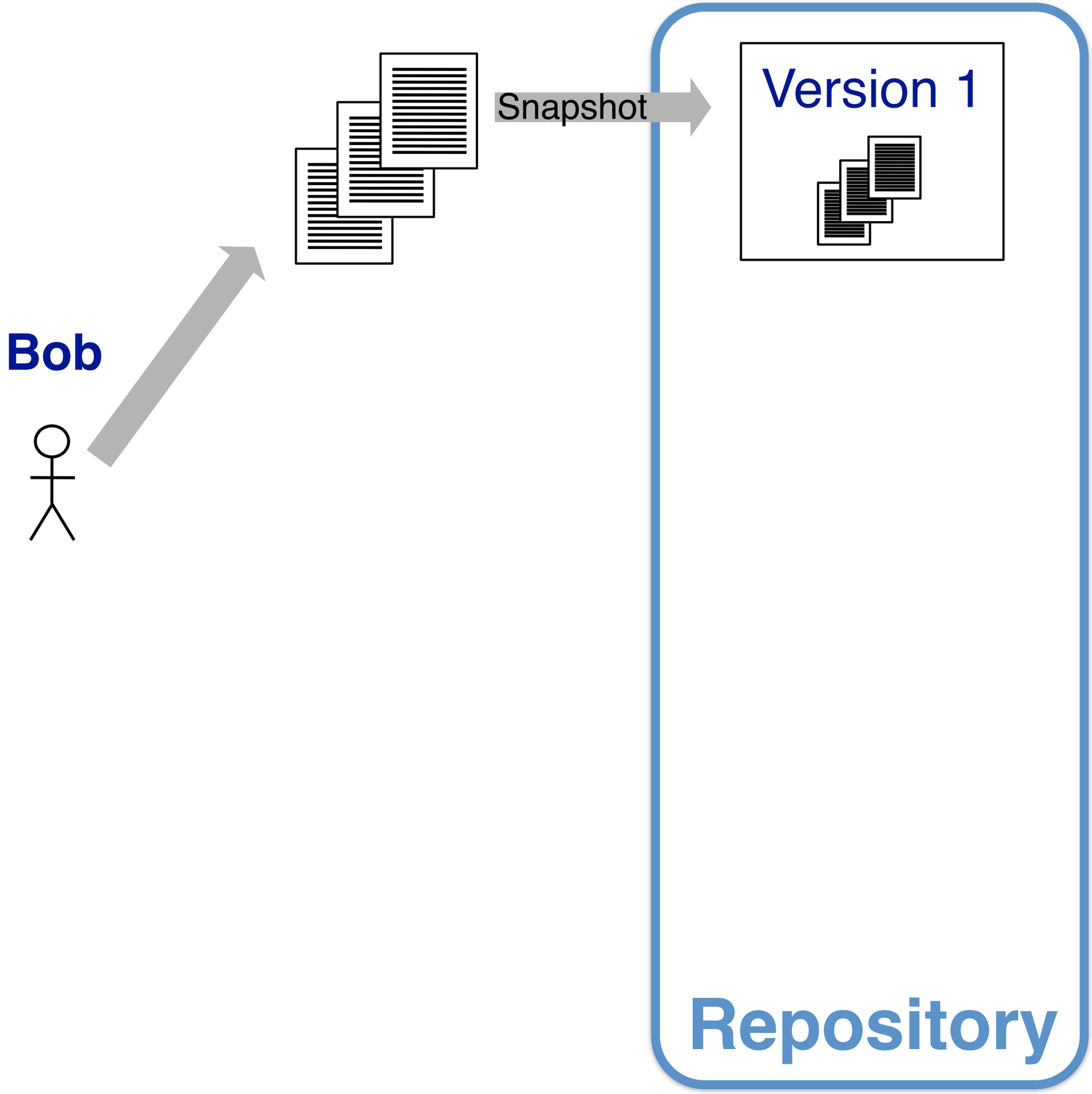
Bob



Repository

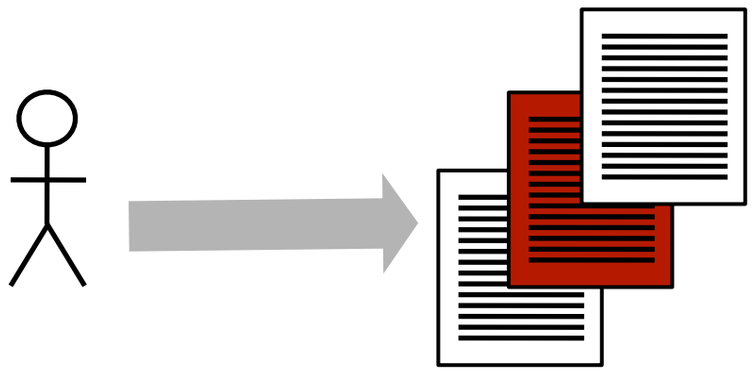

Bob


Repository

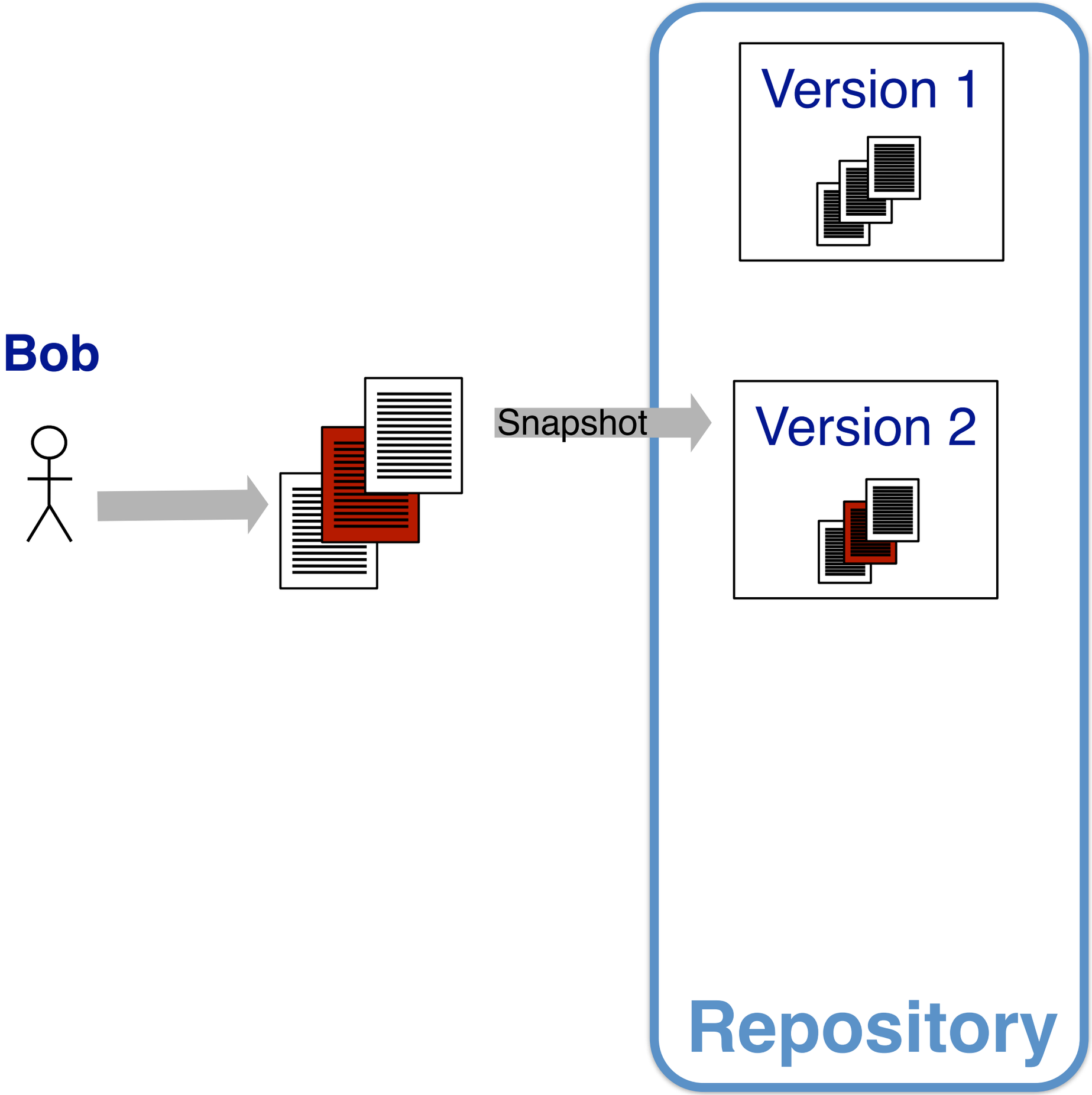

Snapshot
Version 1
Bob
Repository

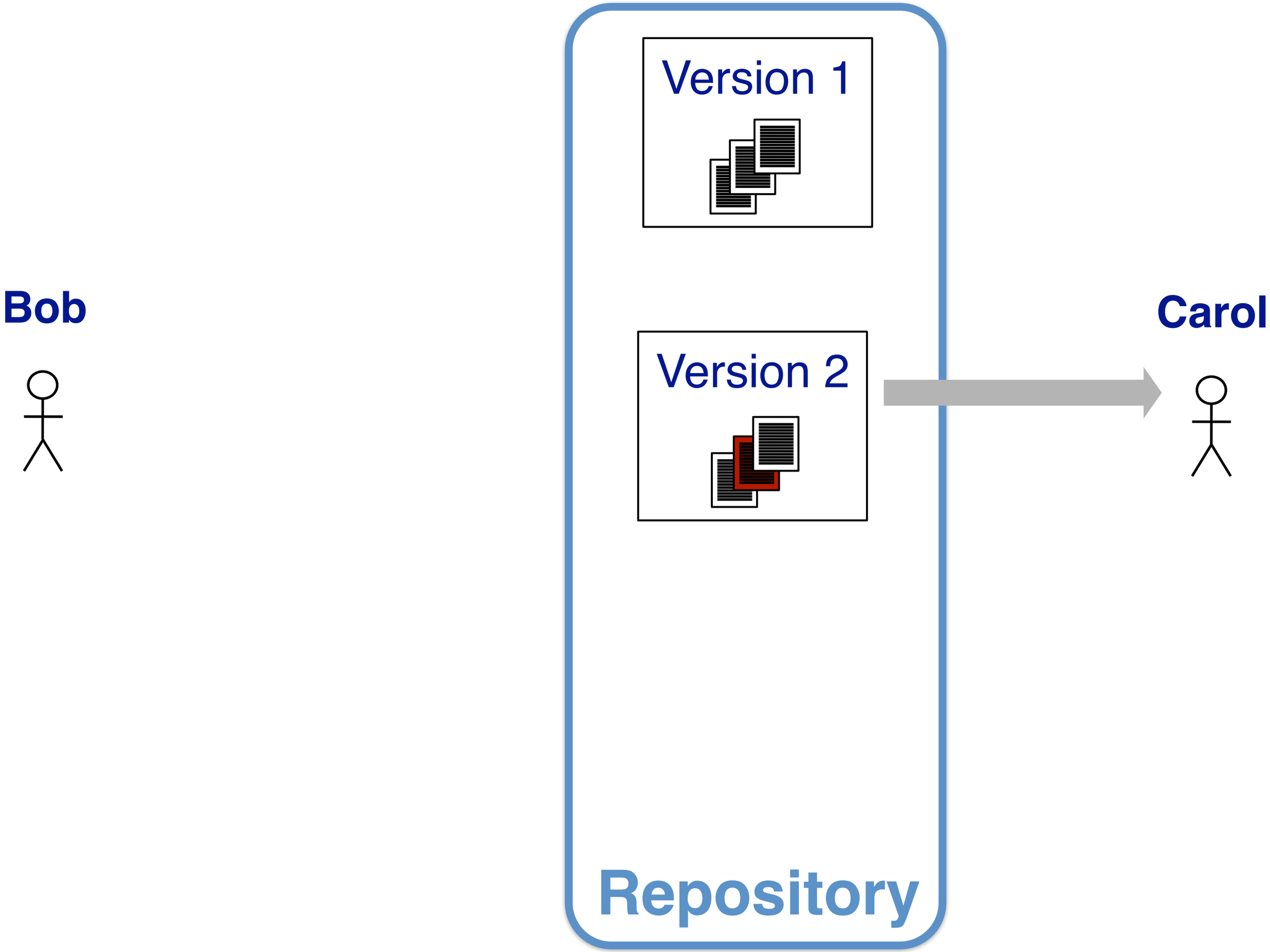
Version 1

Bob

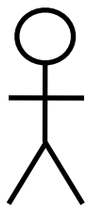


Repository

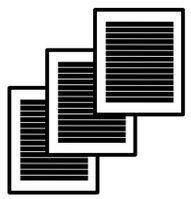

Bob
Snapshot
Version 1
Version 2
Repository


Version 1
Version 2
Bob
Carol
Repository

Bob



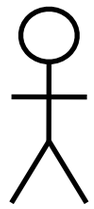
Version 1



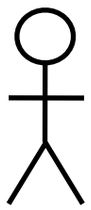
Version 2

Repository

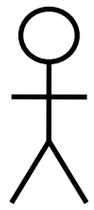
Carol



Bob

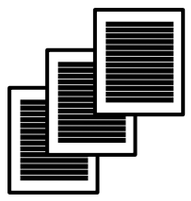


Carol

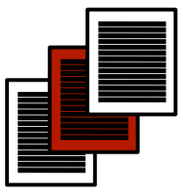


Snapshot

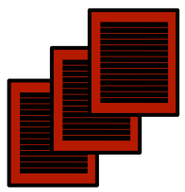
Version 1



Version 2



Version 3

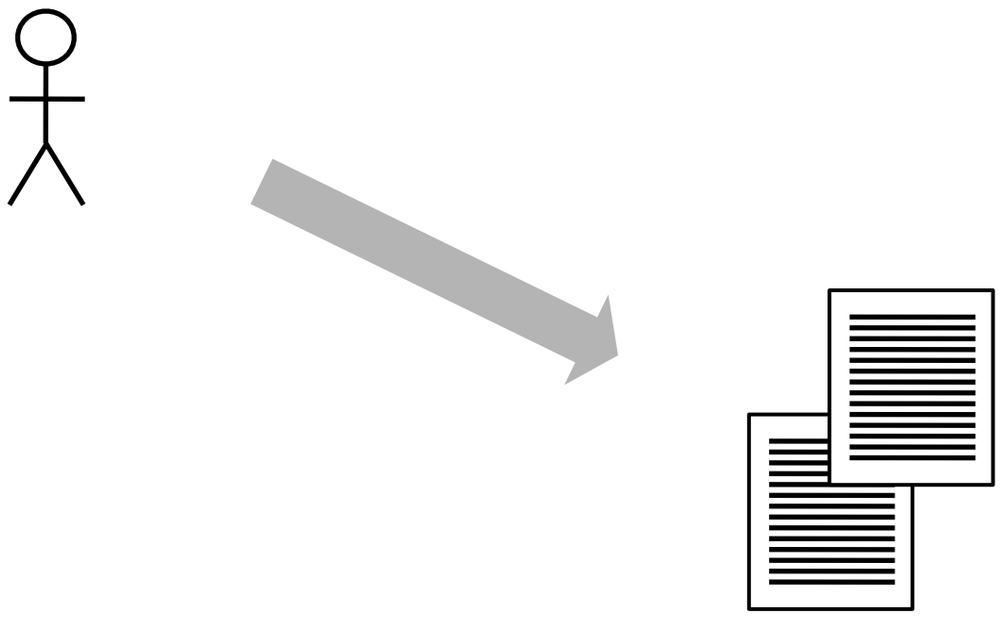


Repository

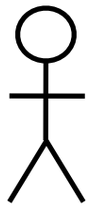
Why use a version control system ?

Cope with the confusion that happens when multiple people edit the same files

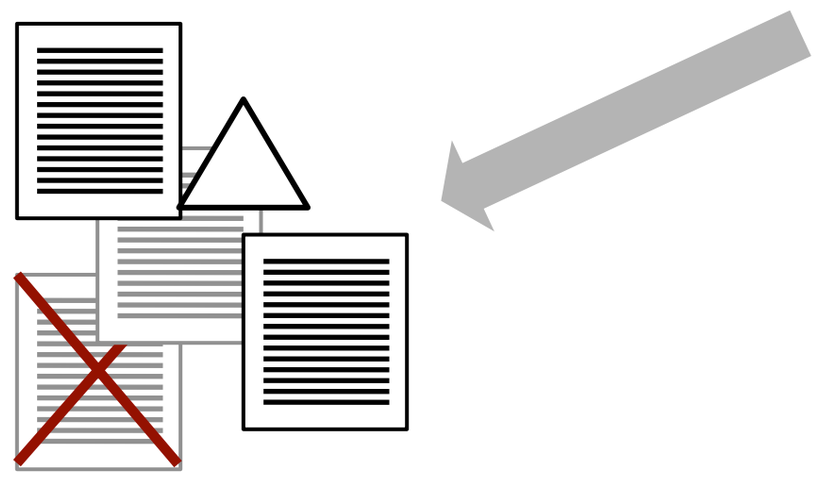
Bob



Bob

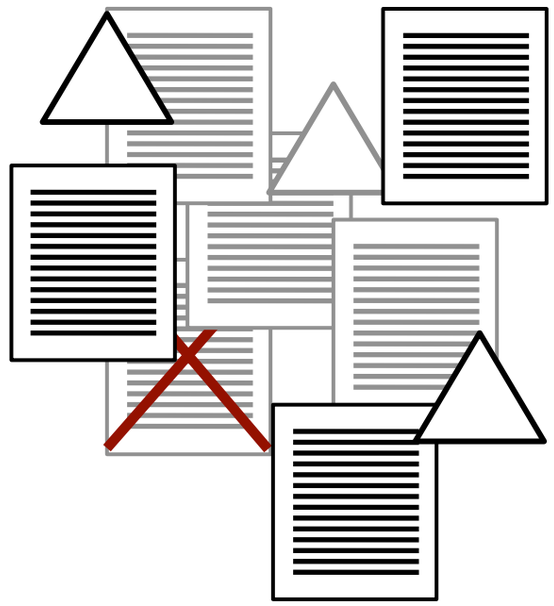


Carol



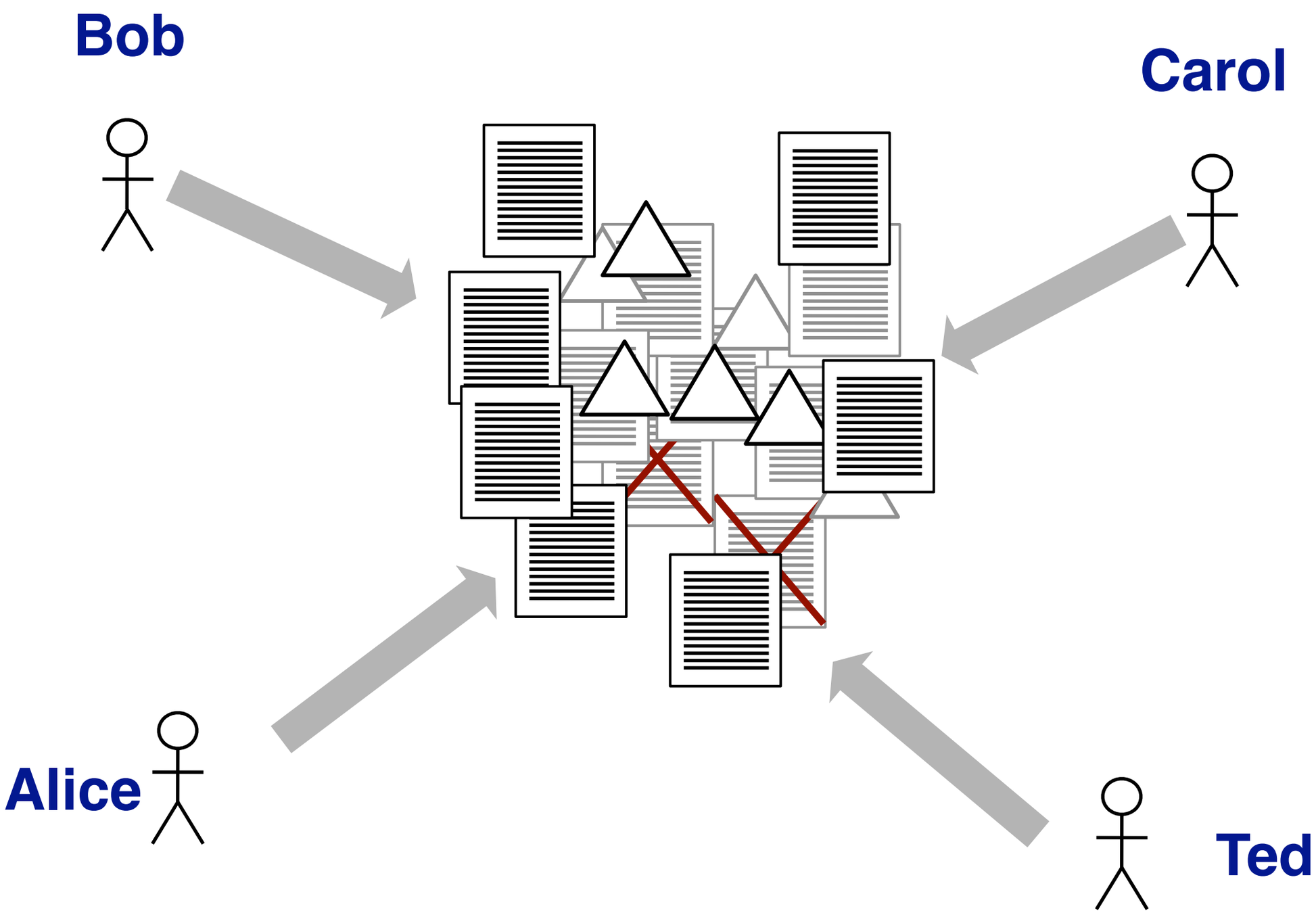
Bob

Carol



Alice

Ted

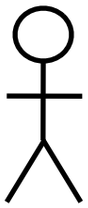


A recipe for disaster!

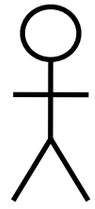
What is a distributed version control system?

What is a centralized version control system?

Bob

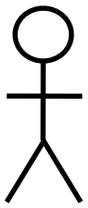


Carol

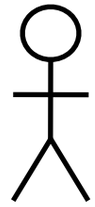


Central repository

Bob

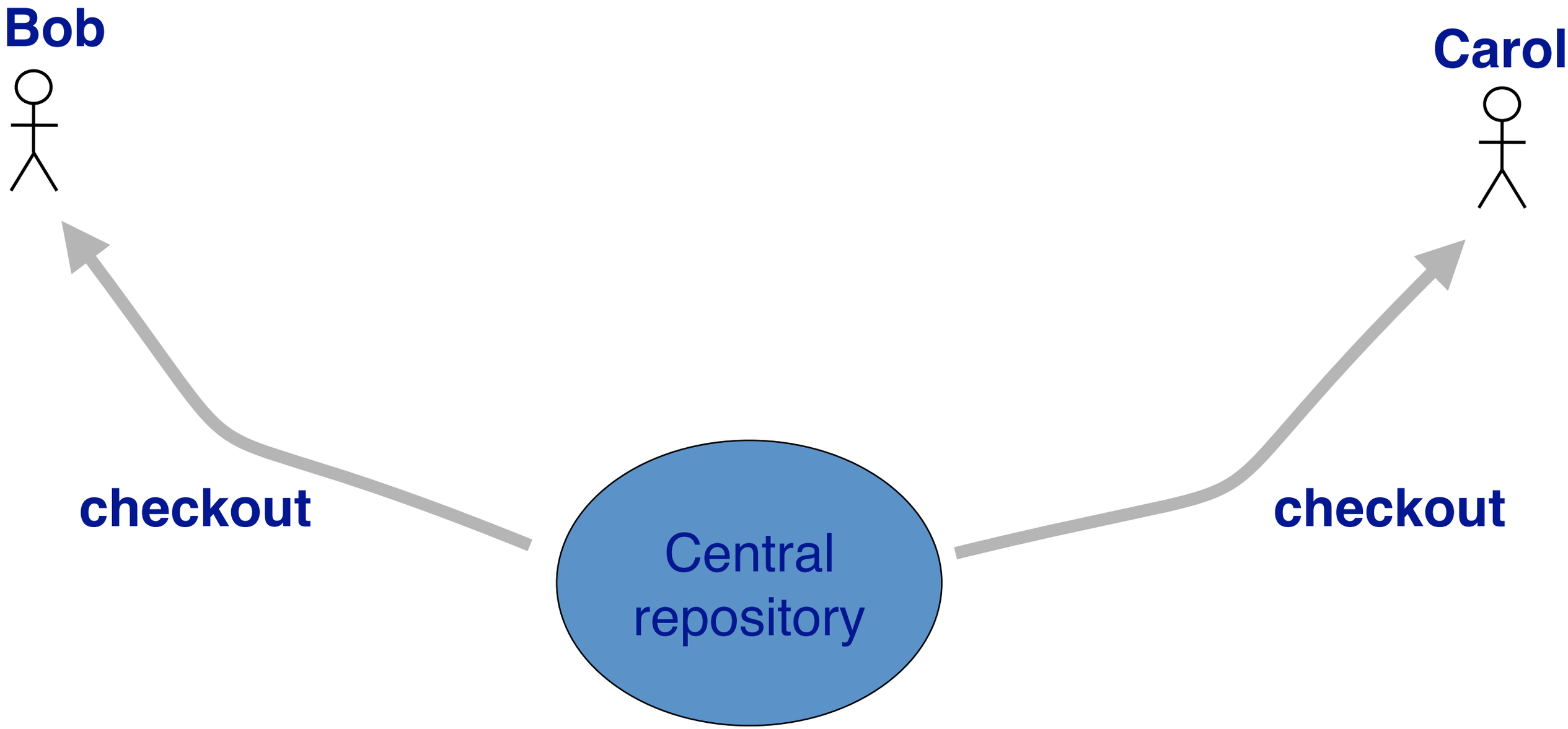


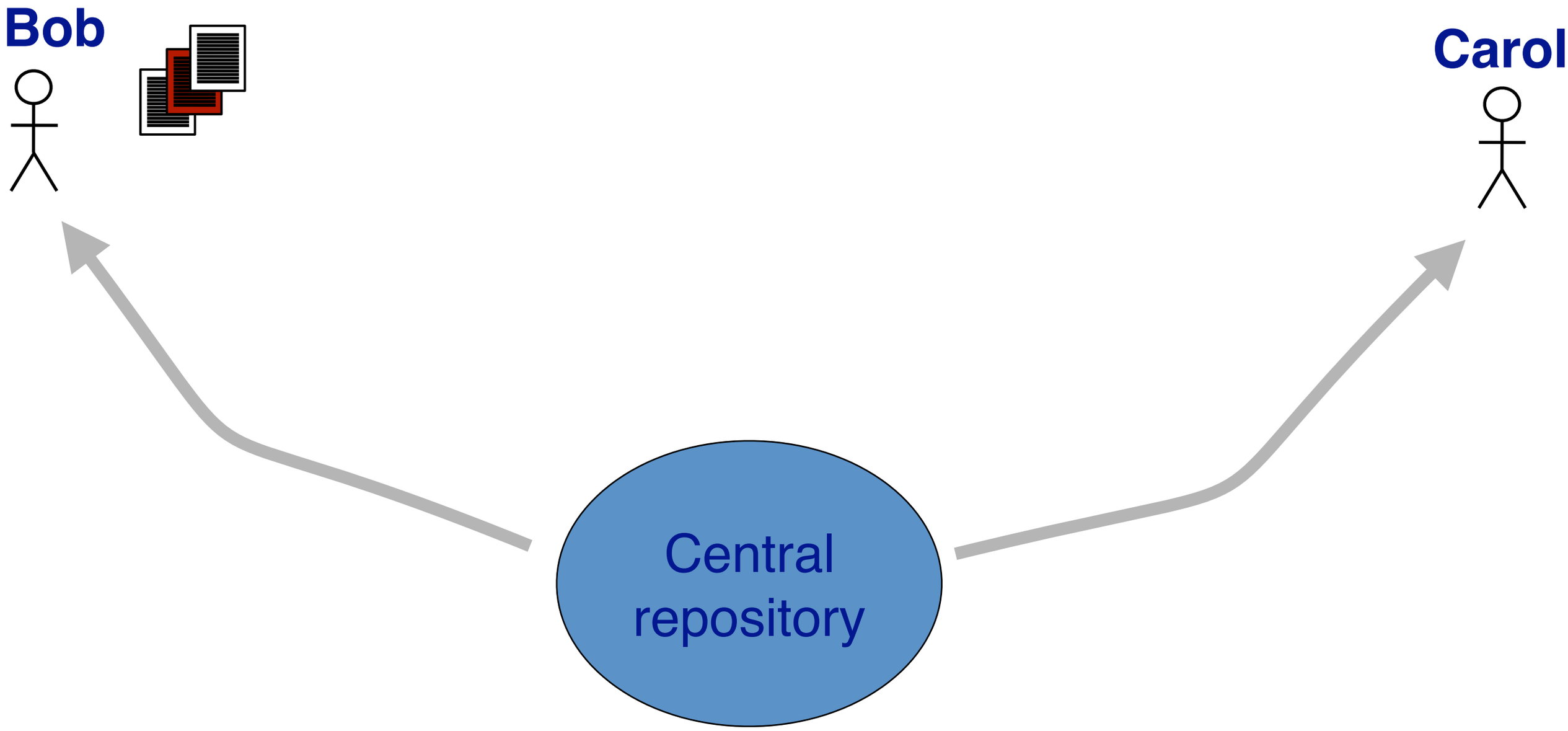
Carol

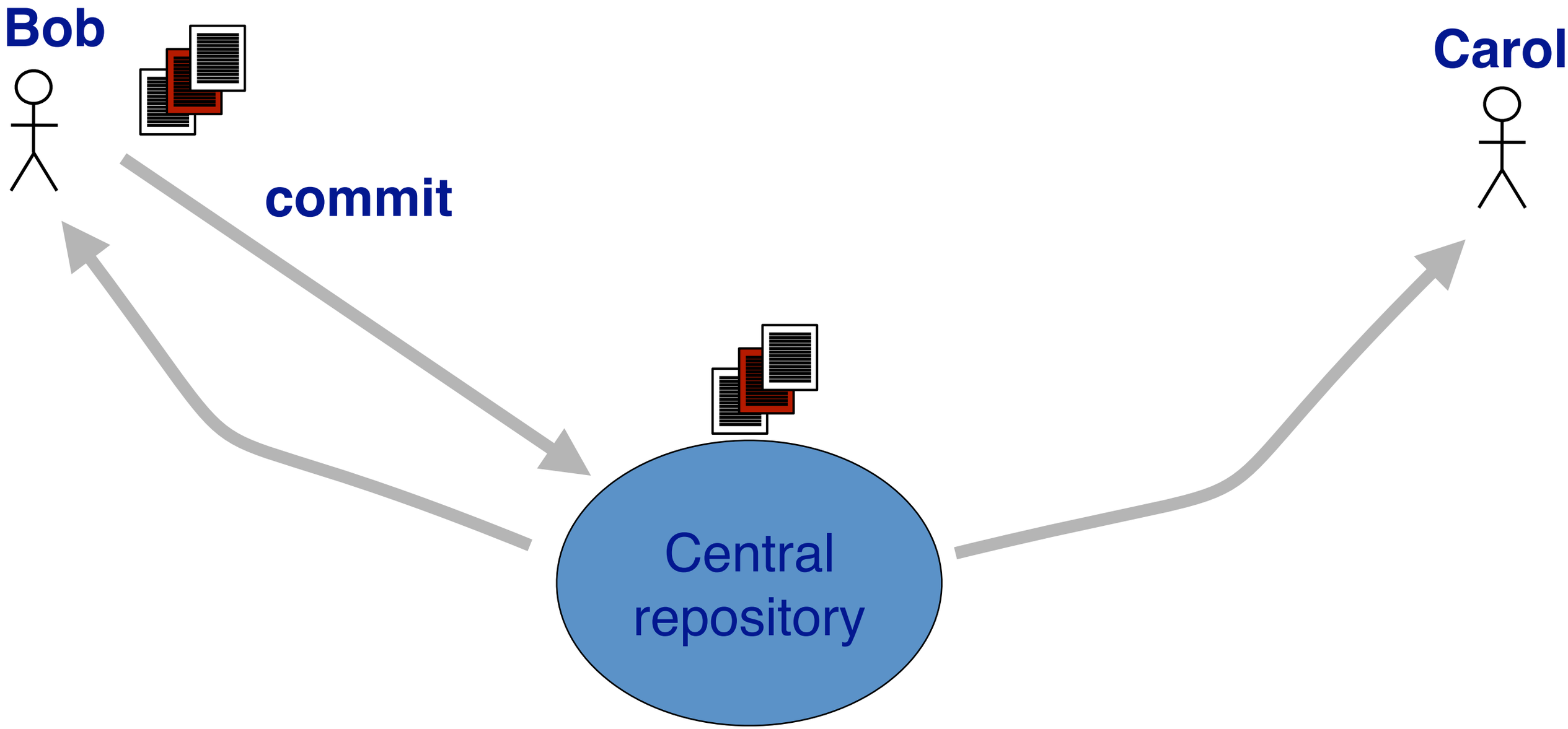


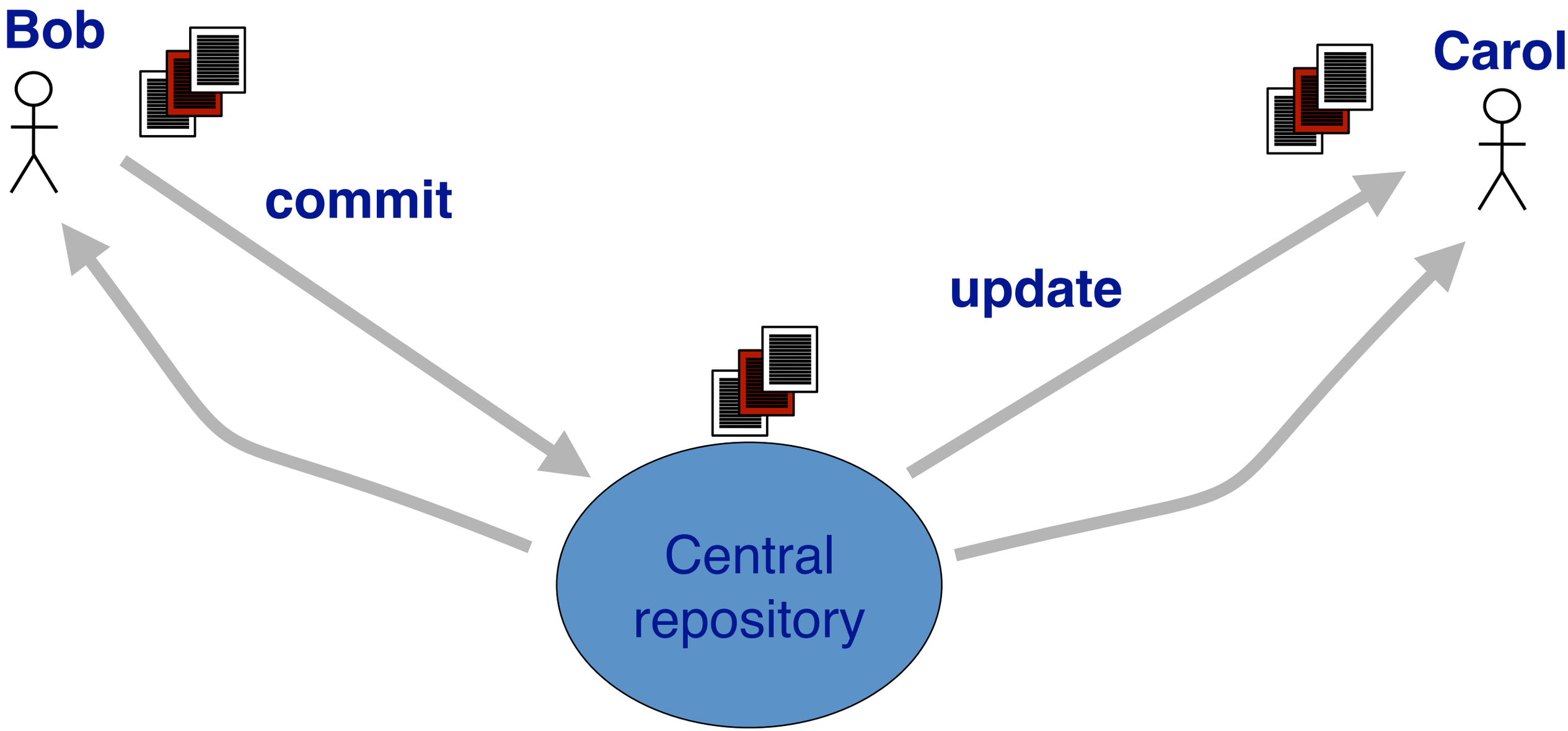
checkout

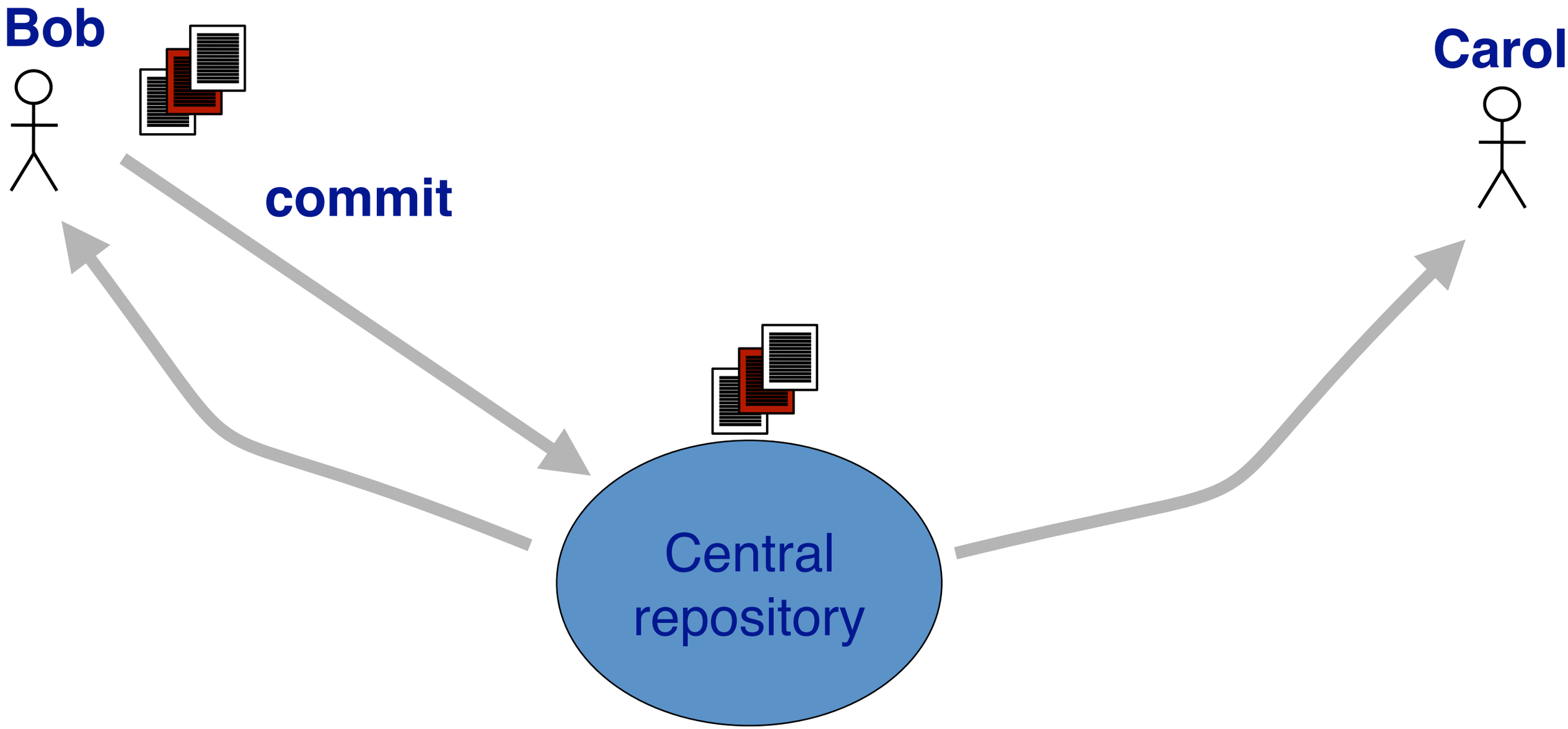
Central repository

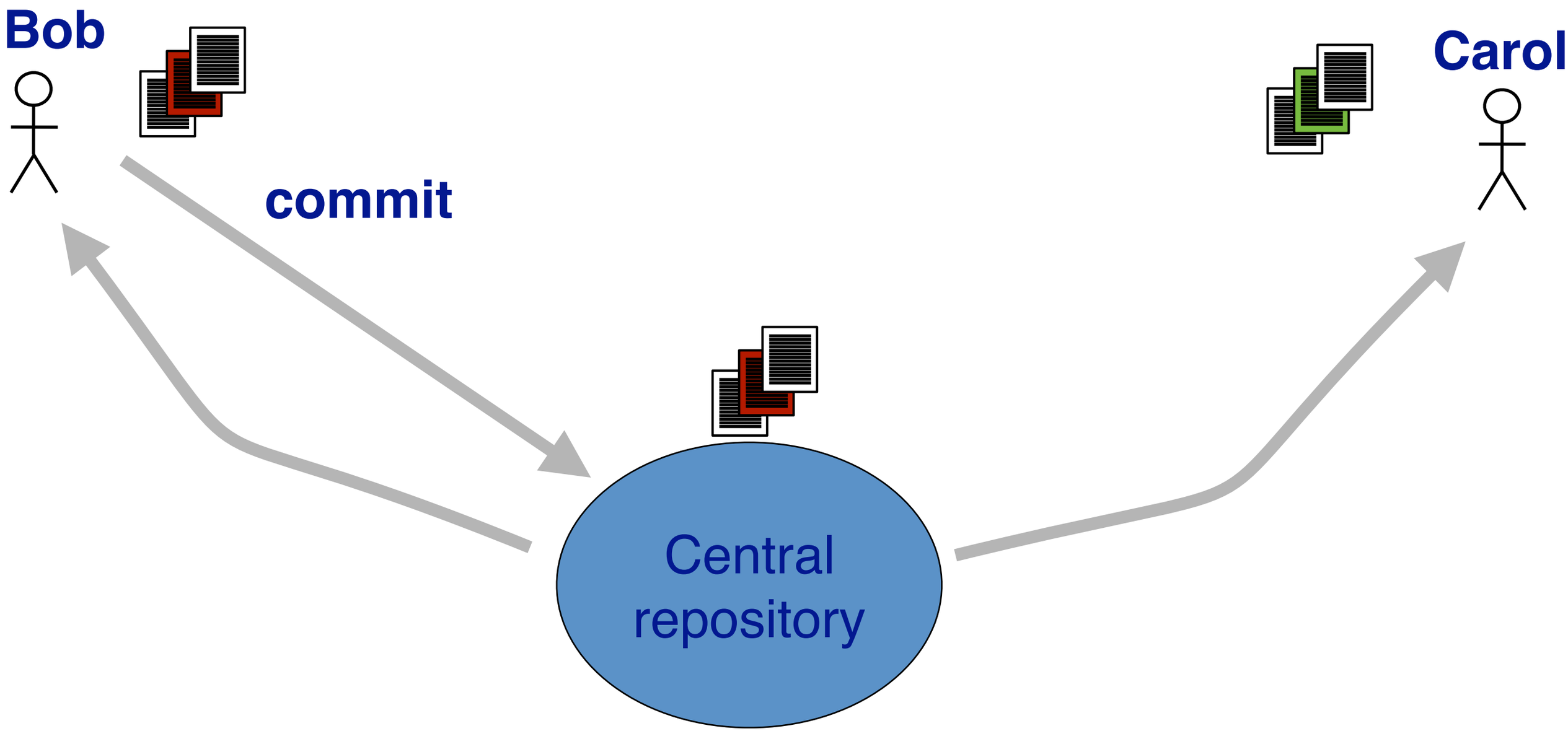

Bob
Carol
checkout
checkout
Central repository

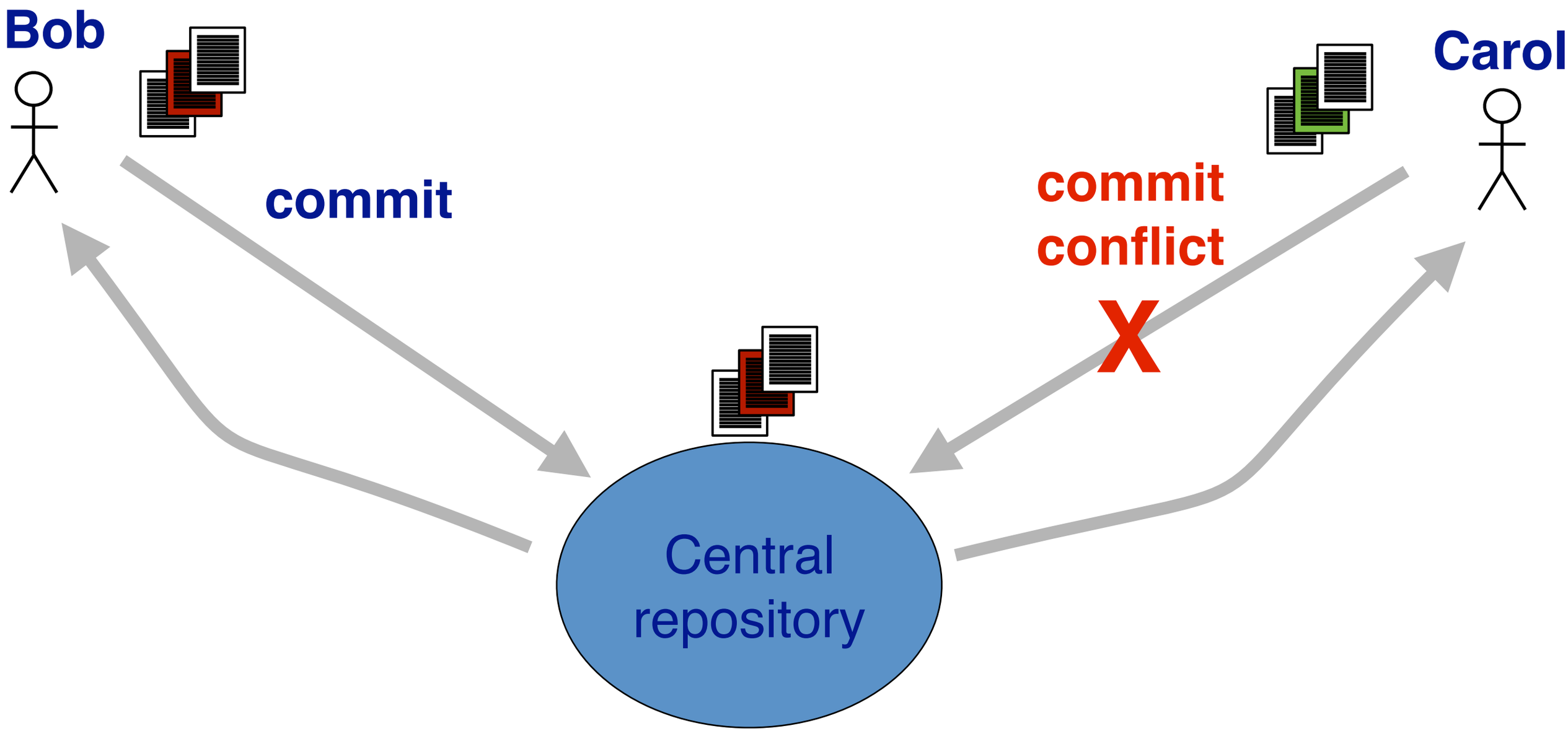

Bob
Carol
Central repository


Bob
Carol
commit
Central
repository


Bob
Carol
commit
update
Central
repository

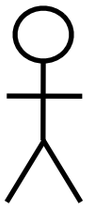

Bob
Carol
commit
Central
repository


Bob
commit
Carol
Central
repository

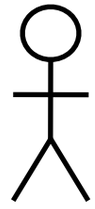

Bob
commit
Carol
commit
conflict
X
Central
repository

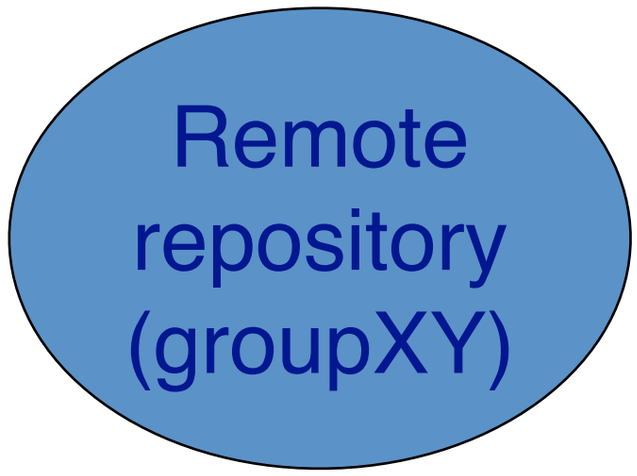
What is a distributed version control system?

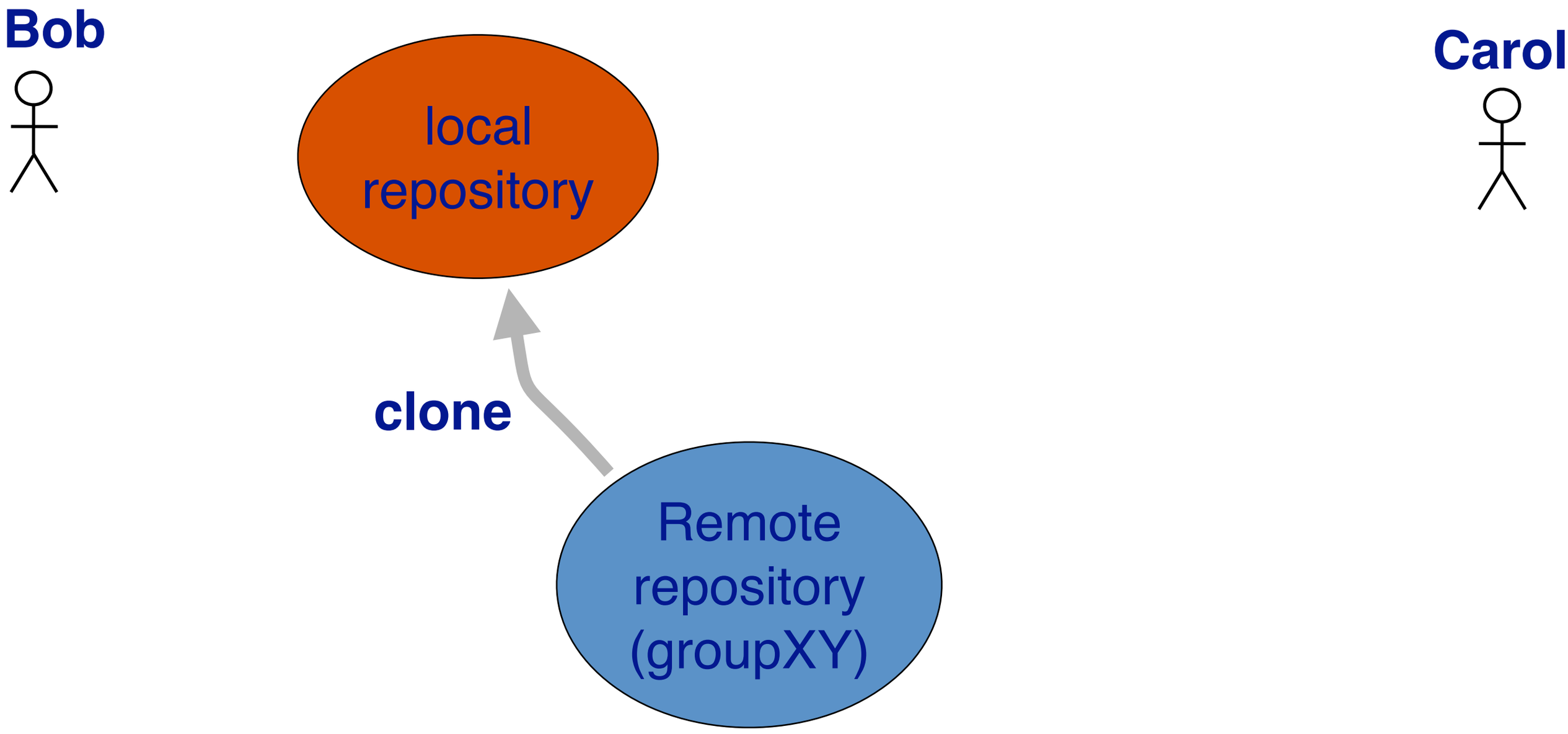
Bob

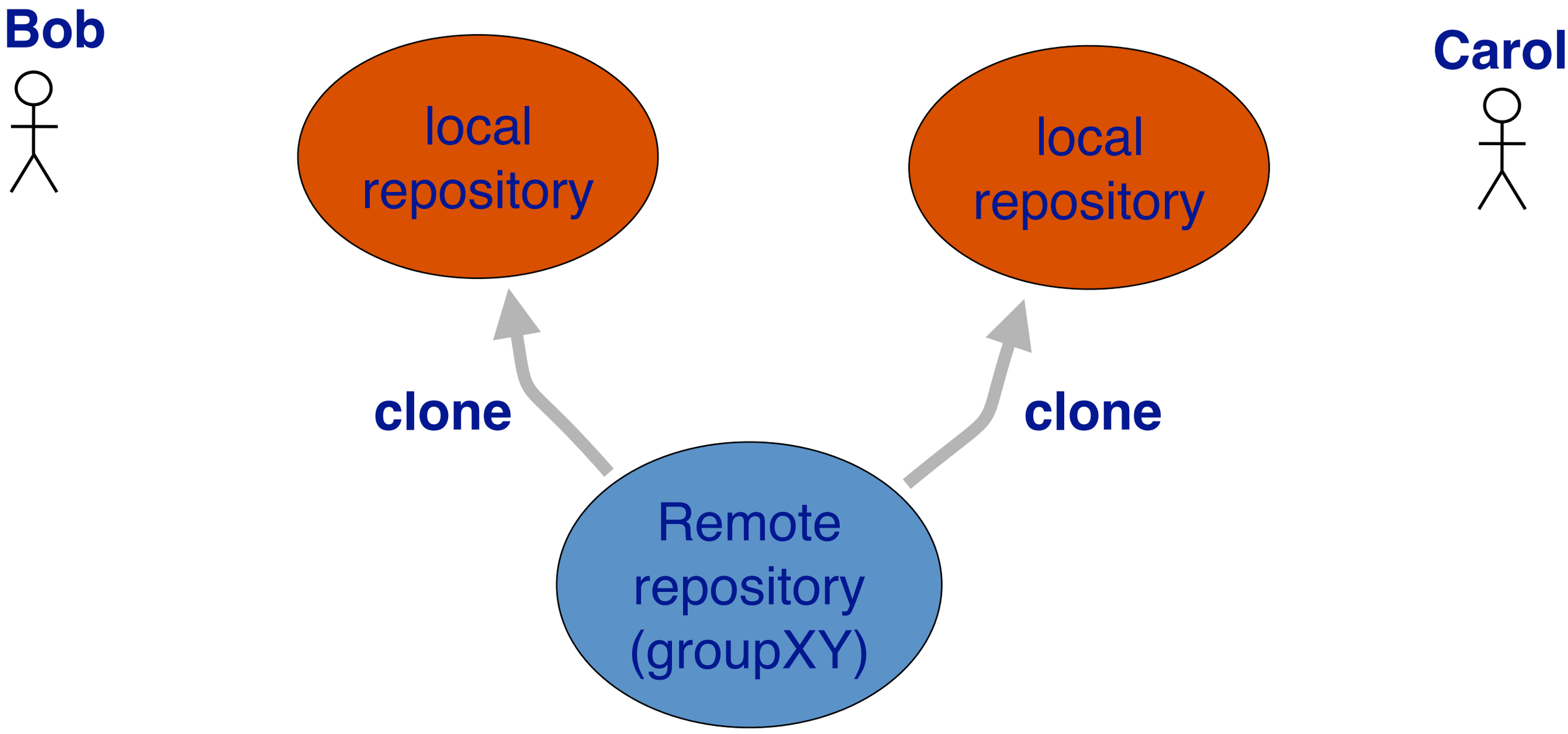


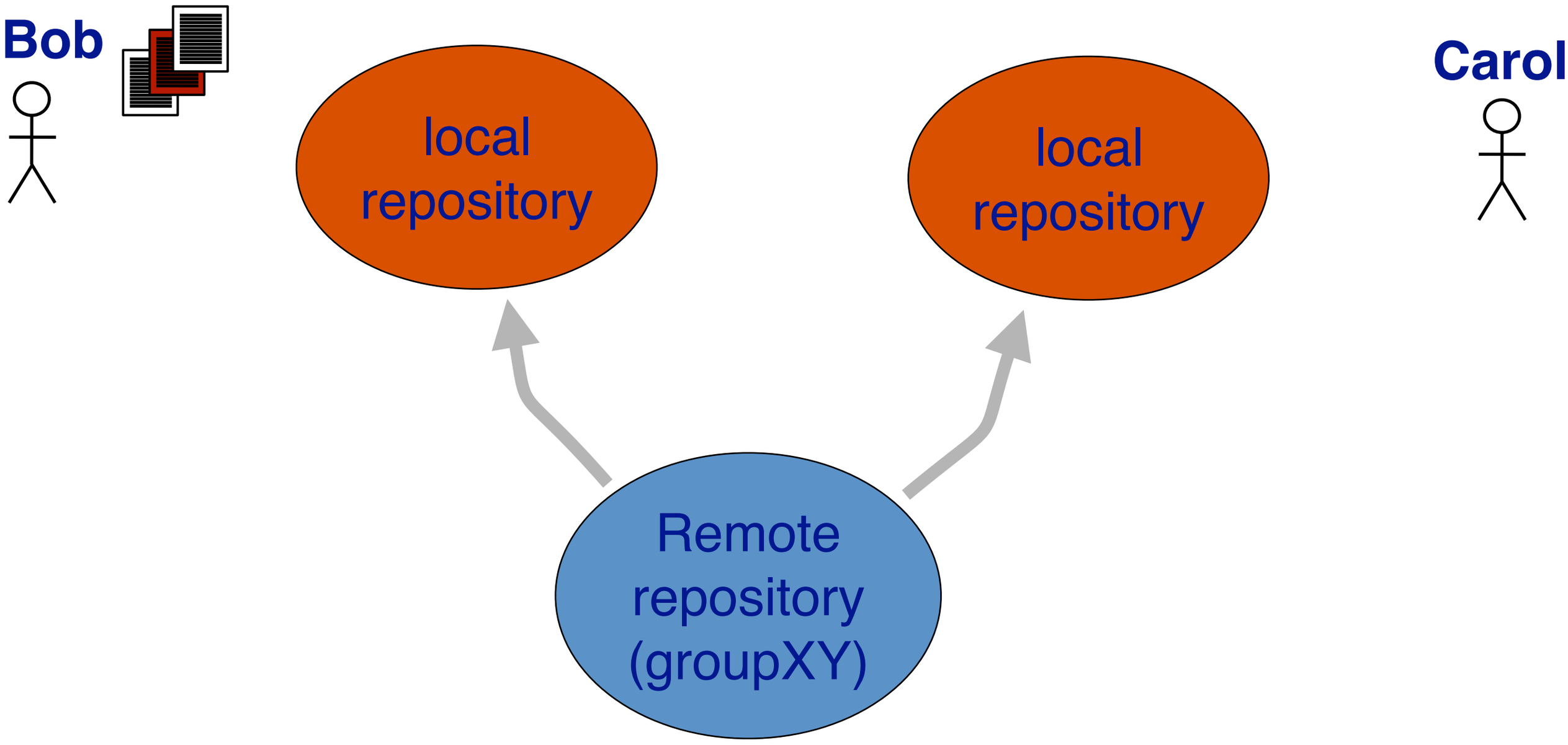
Carol

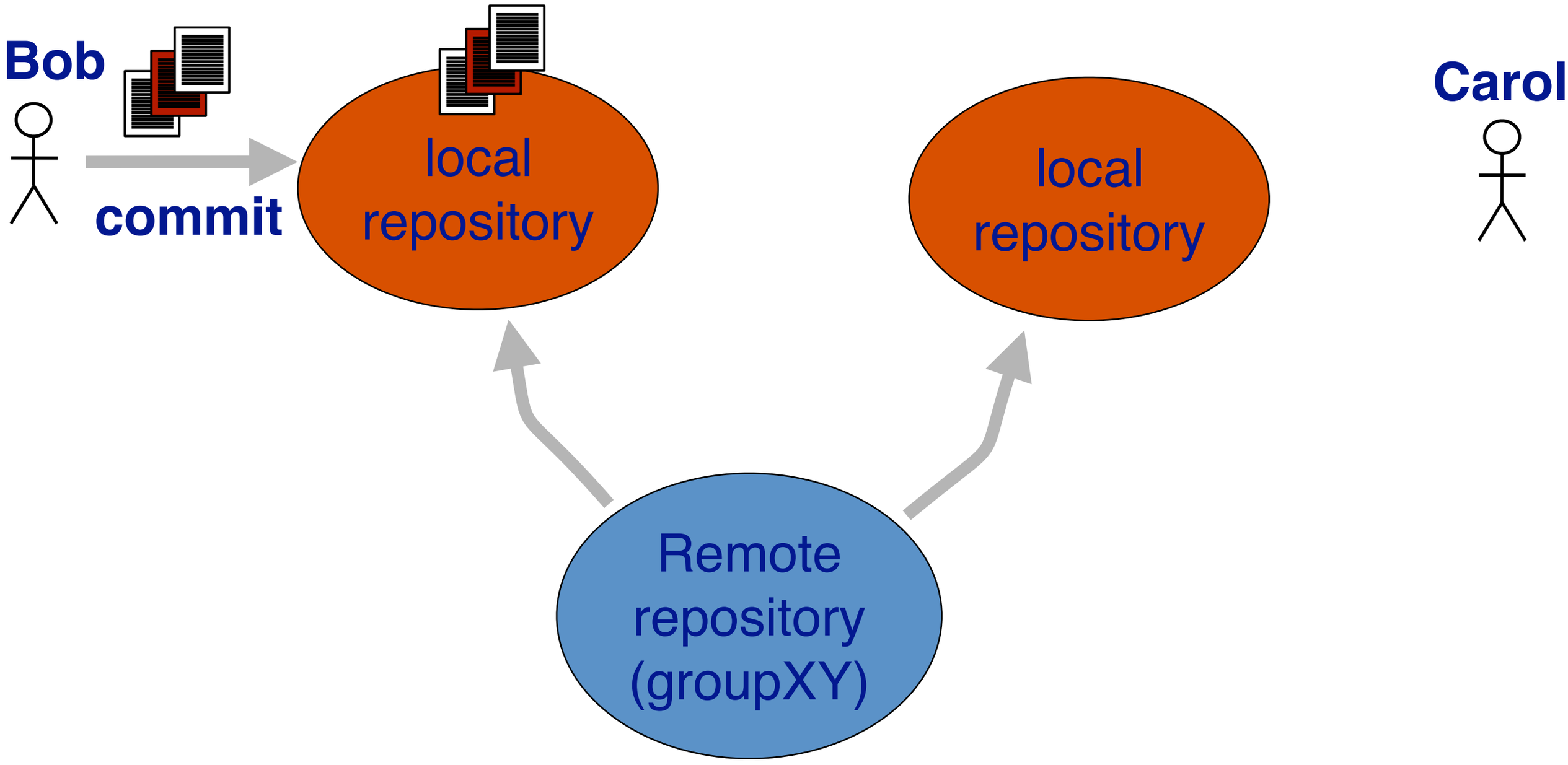


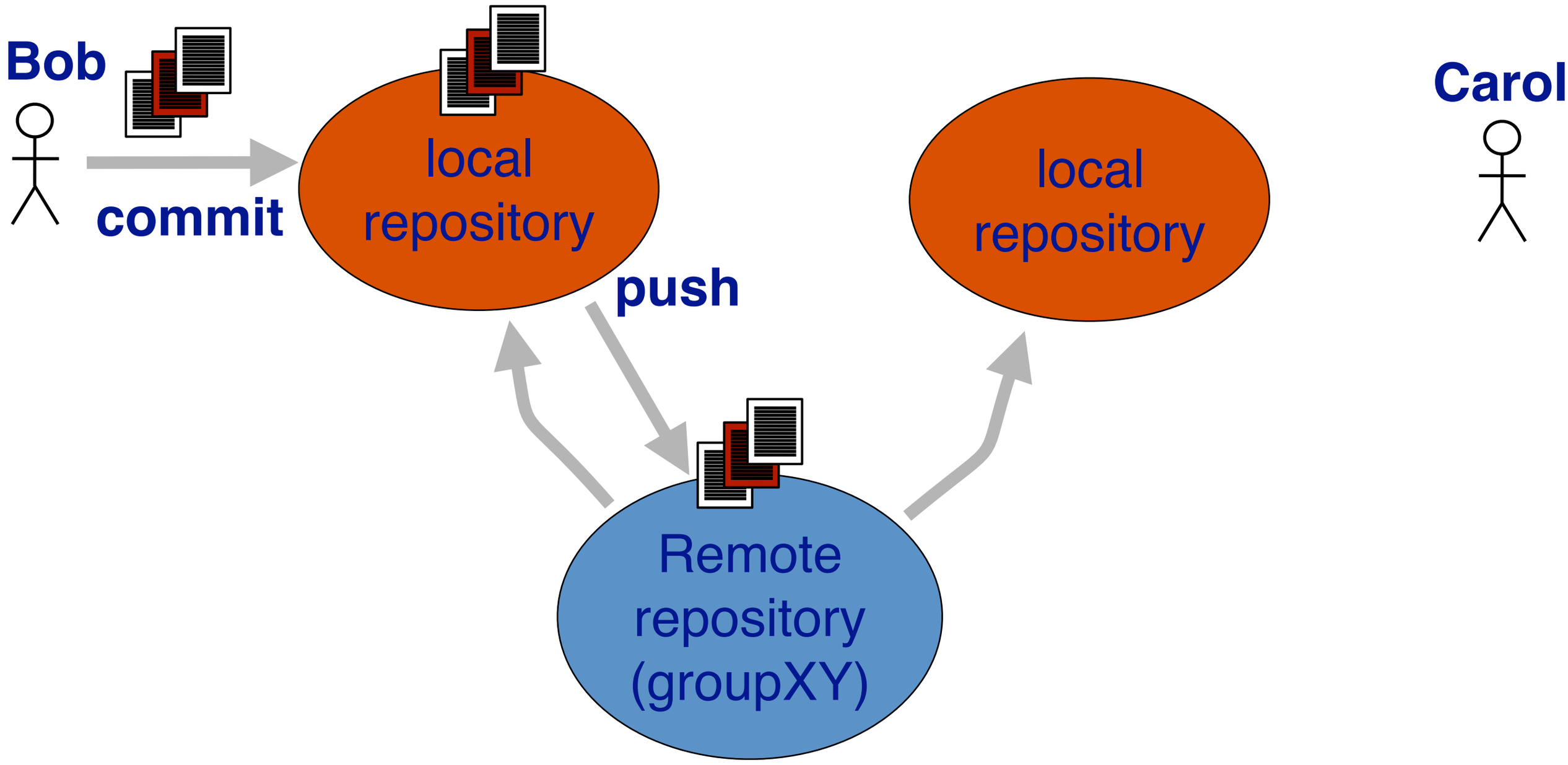


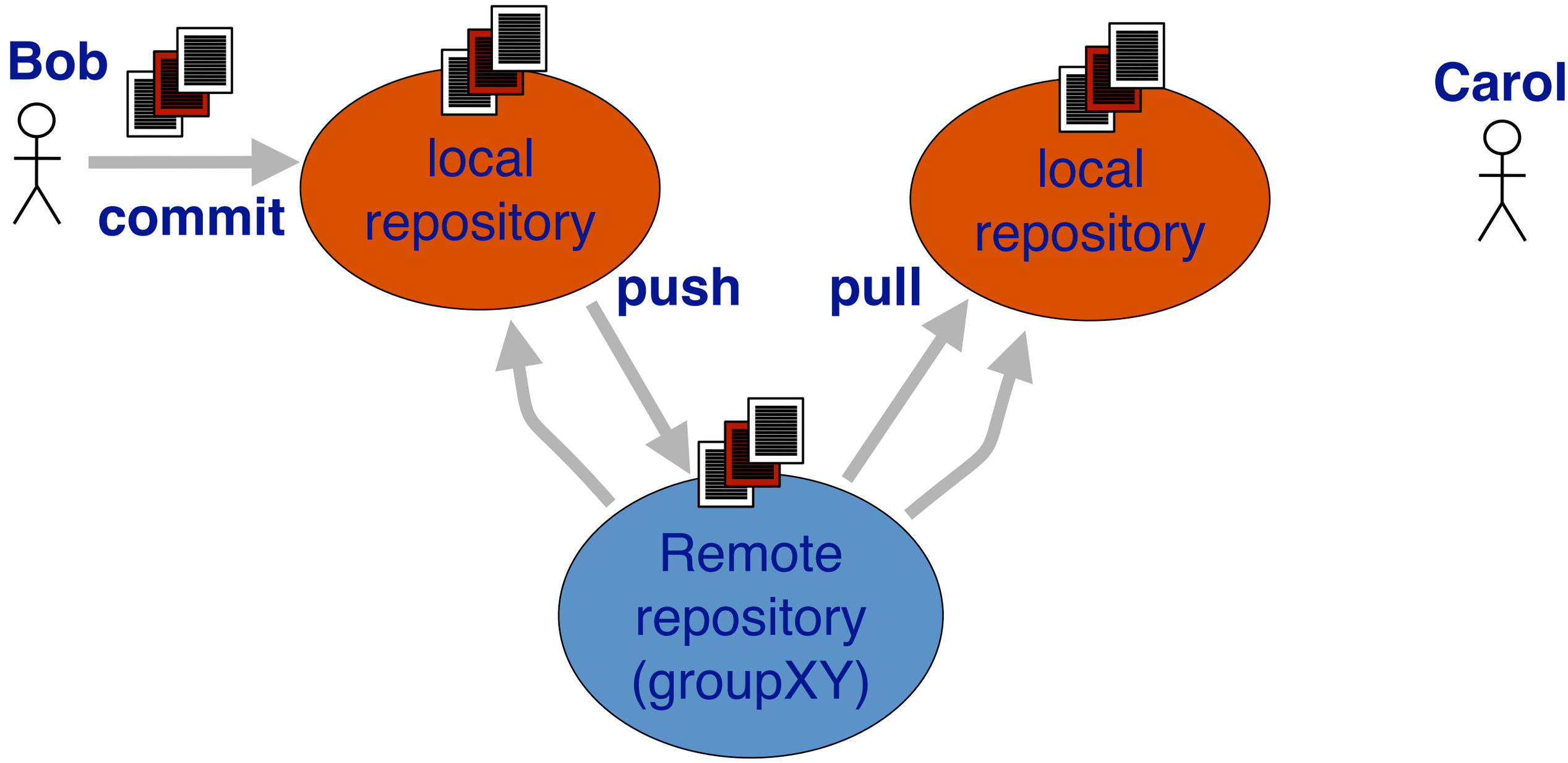

Bob
local
repository
Carol
clone
Remote
repository
(groupXY)

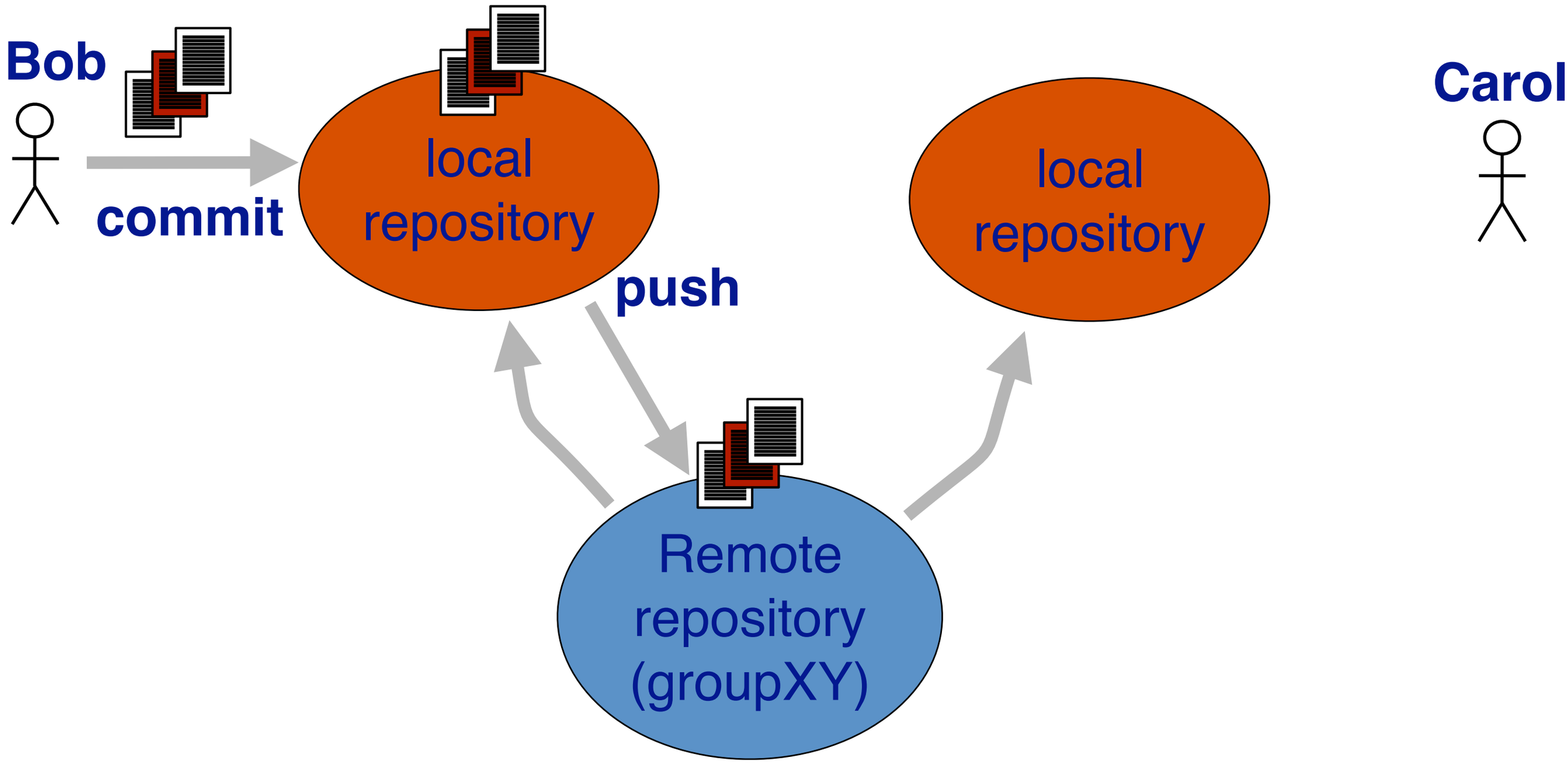

Bob
Carol
local repository
local repository
clone
clone
Remote repository (groupXY)

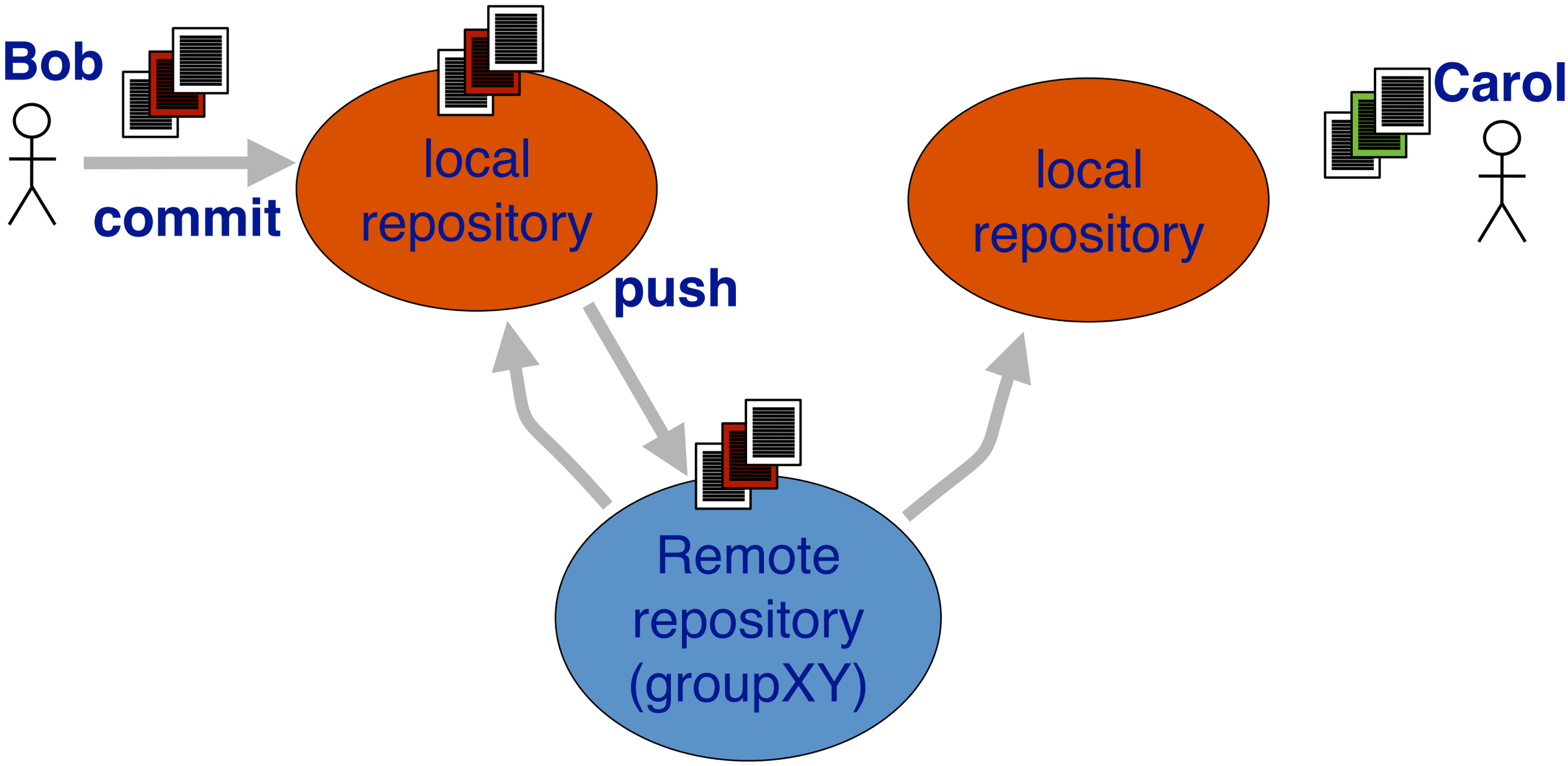

Bob
local repository
local repository
Carol
Remote repository (groupXY)

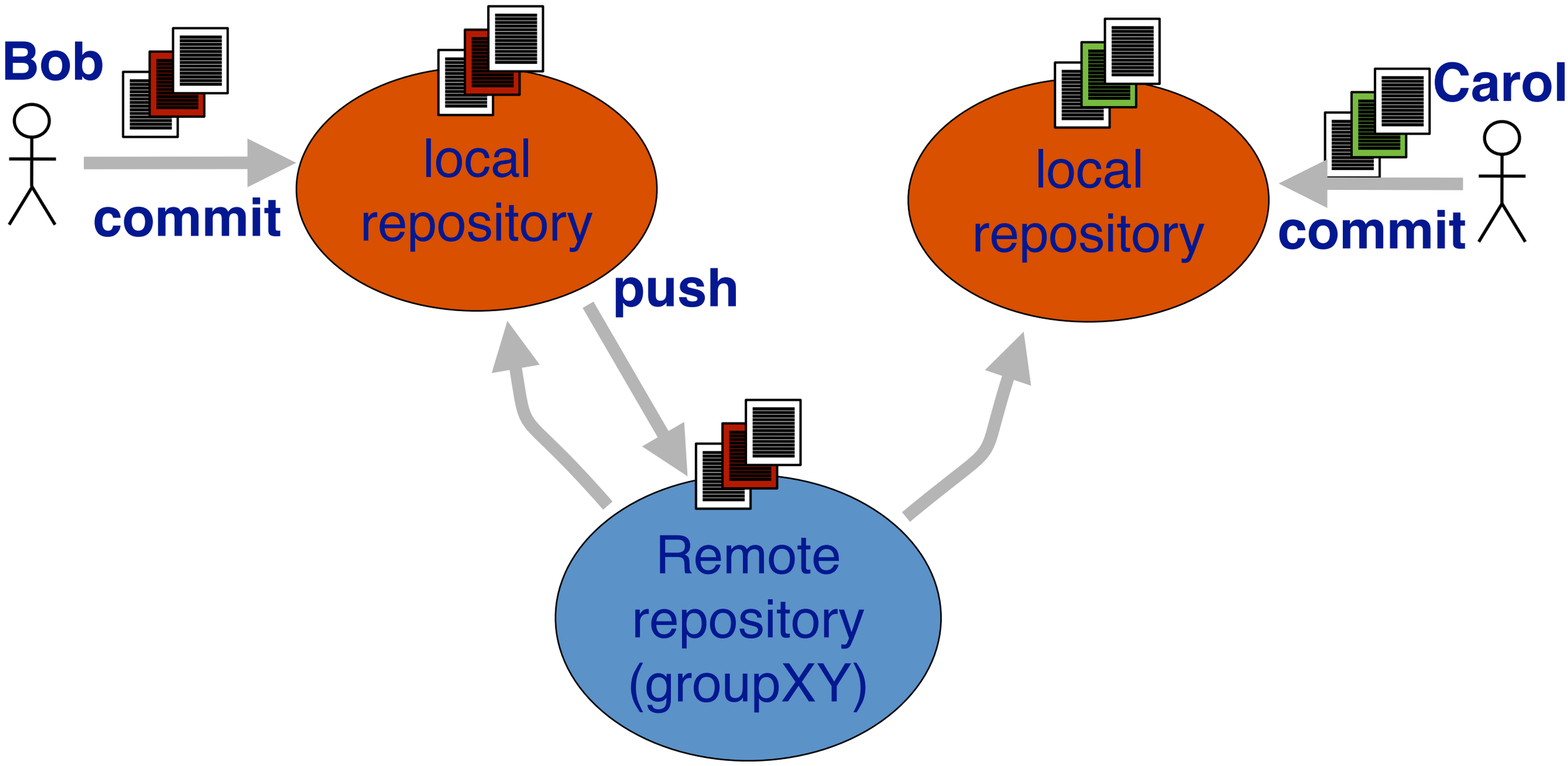

Bob
commit
local repository
local repository
Carol
Remote repository (groupXY)

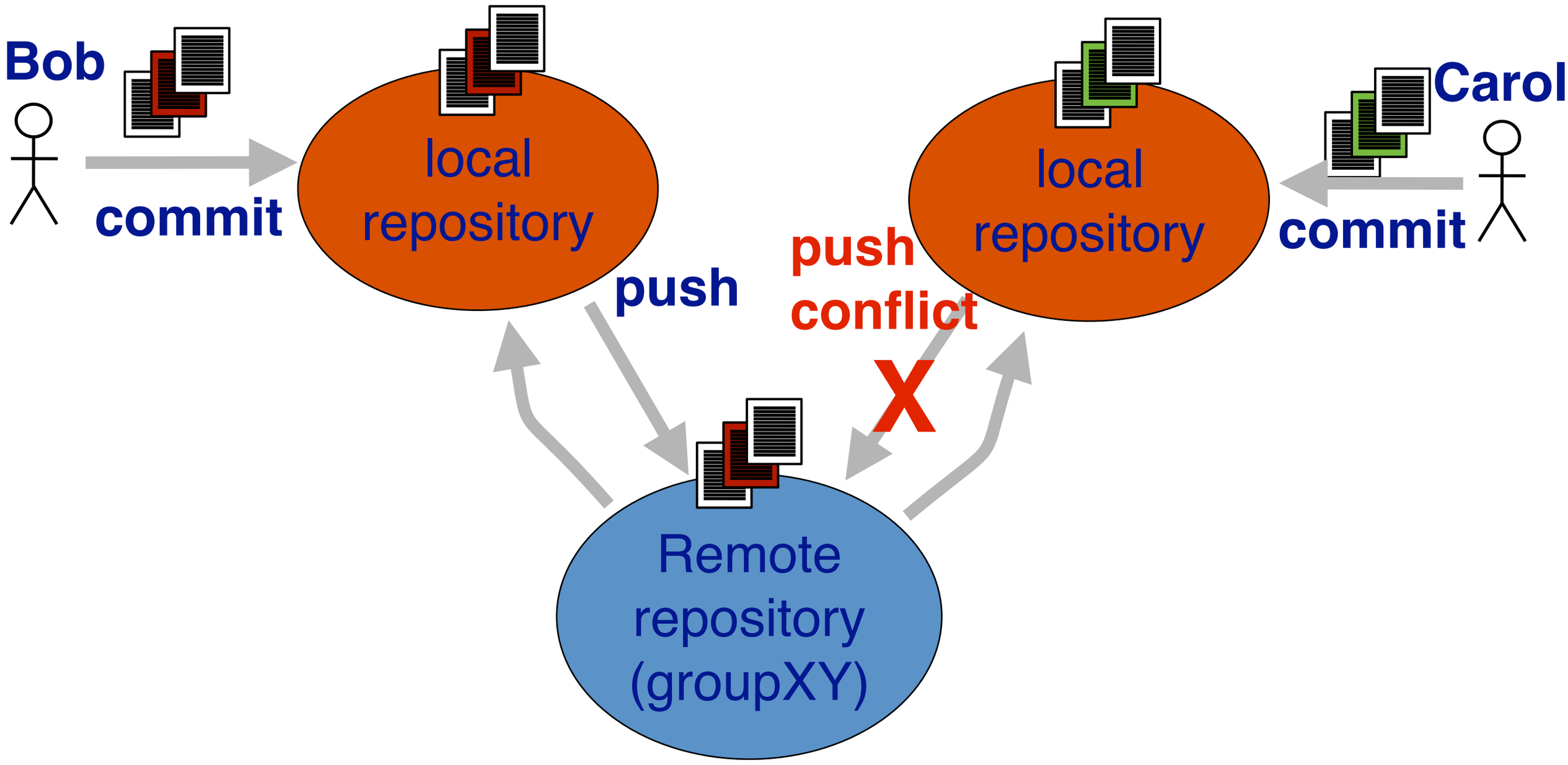

Bob
commit
local
repository
push
local
repository
Carol
Remote
repository
(groupXY)


Bob
commit
local repository
push
pull
local repository
Carol
Remote repository (groupXY)


Bob
commit
local
repository
push
local
repository
Carol
Remote
repository
(groupXY)


Bob
commit
local
repository
push
local
repository
Carol
Remote
repository
(groupXY)


Bob
commit
local
repository
push
local
repository
Carol
commit
Remote
repository
(groupXY)


Bob
commit
local repository
push
push conflict
X
local repository
Carol
commit
Remote repository (groupXY)

Why git?

Distributed version control system

Good tool support

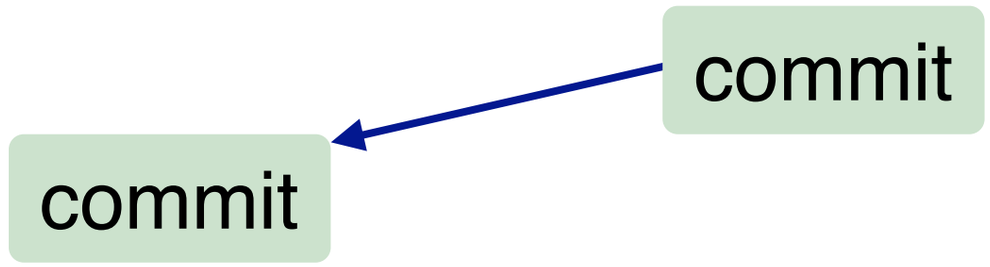
Fast/stable

Basic git

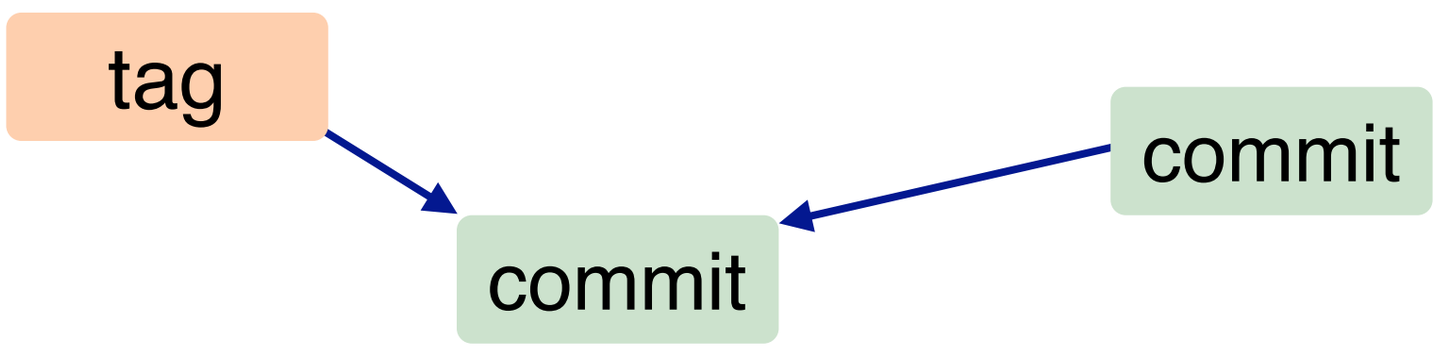
A “commit” is
“a set of changes”
to a “set of files”

commit

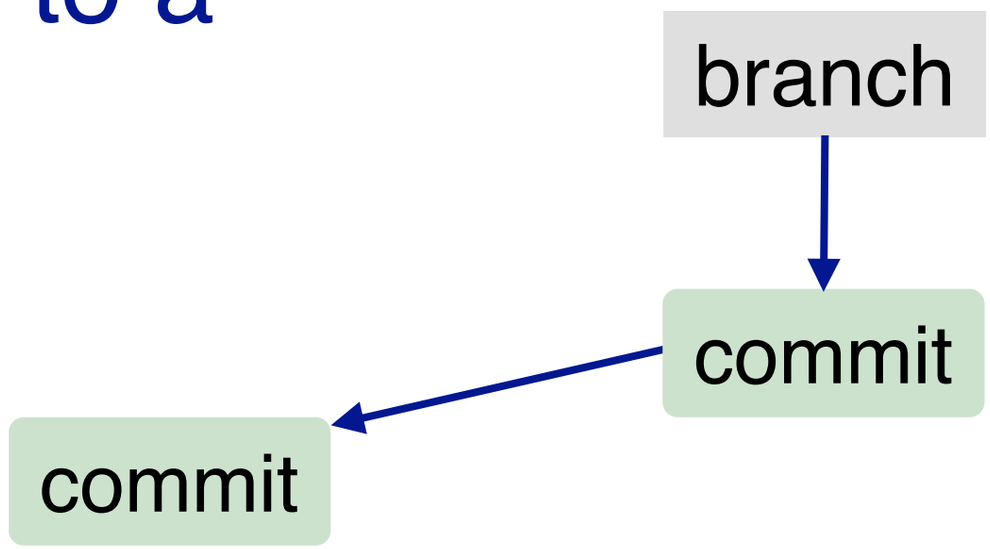
Most commits
modify (or merge)
earlier commits



You can “tag” an interesting commit

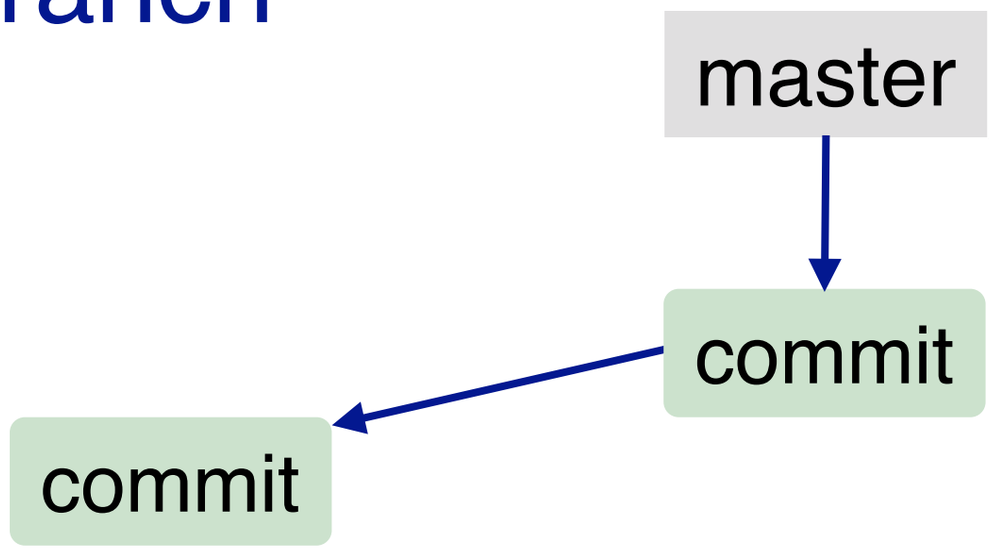


A graph of commits may belong to a *branch*

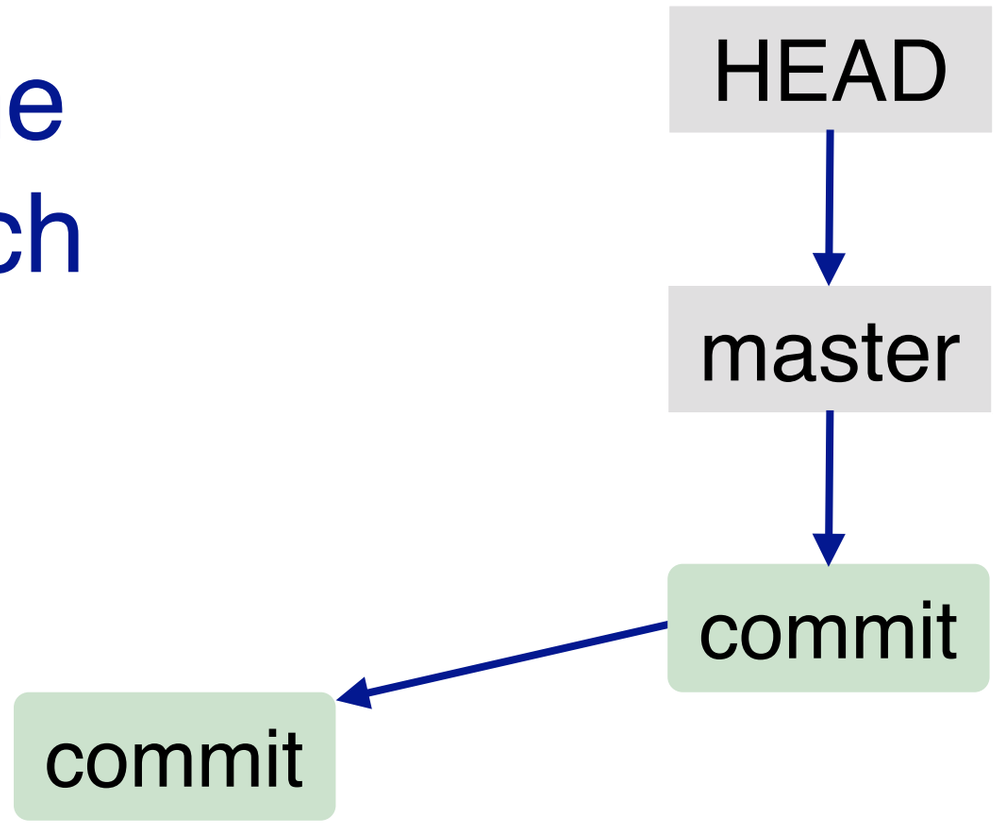


master

is the main branch

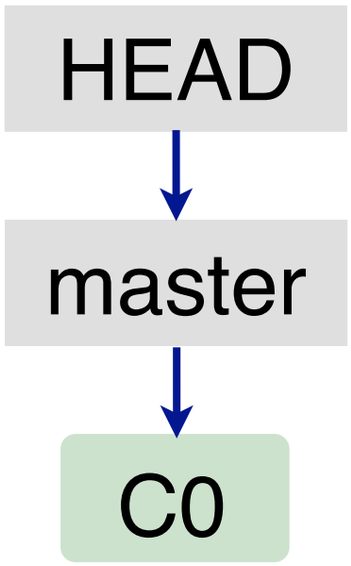


“HEAD “is the current branch

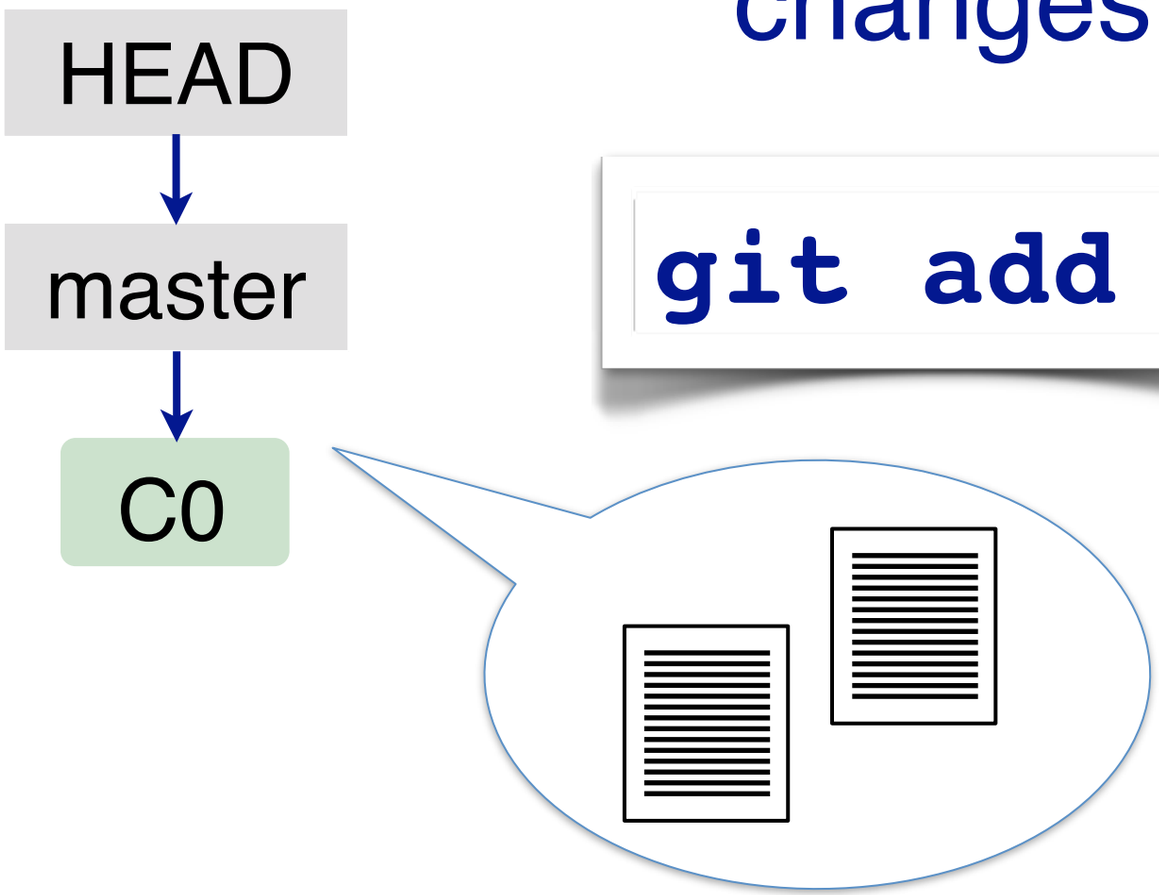


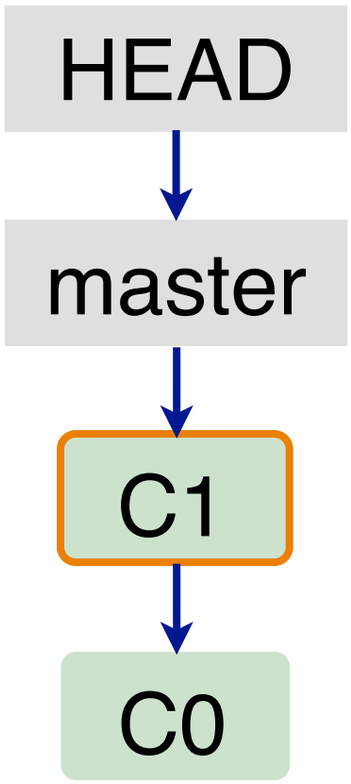
Create a git repo

```
mkdir repo
cd repo
git init
```



Tell git to “stage” changes

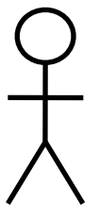




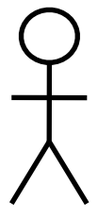
Commit your changes

```
git commit …
```

Collaborating



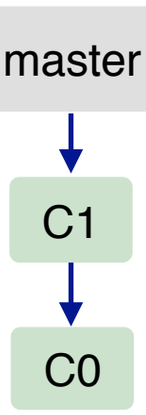
John



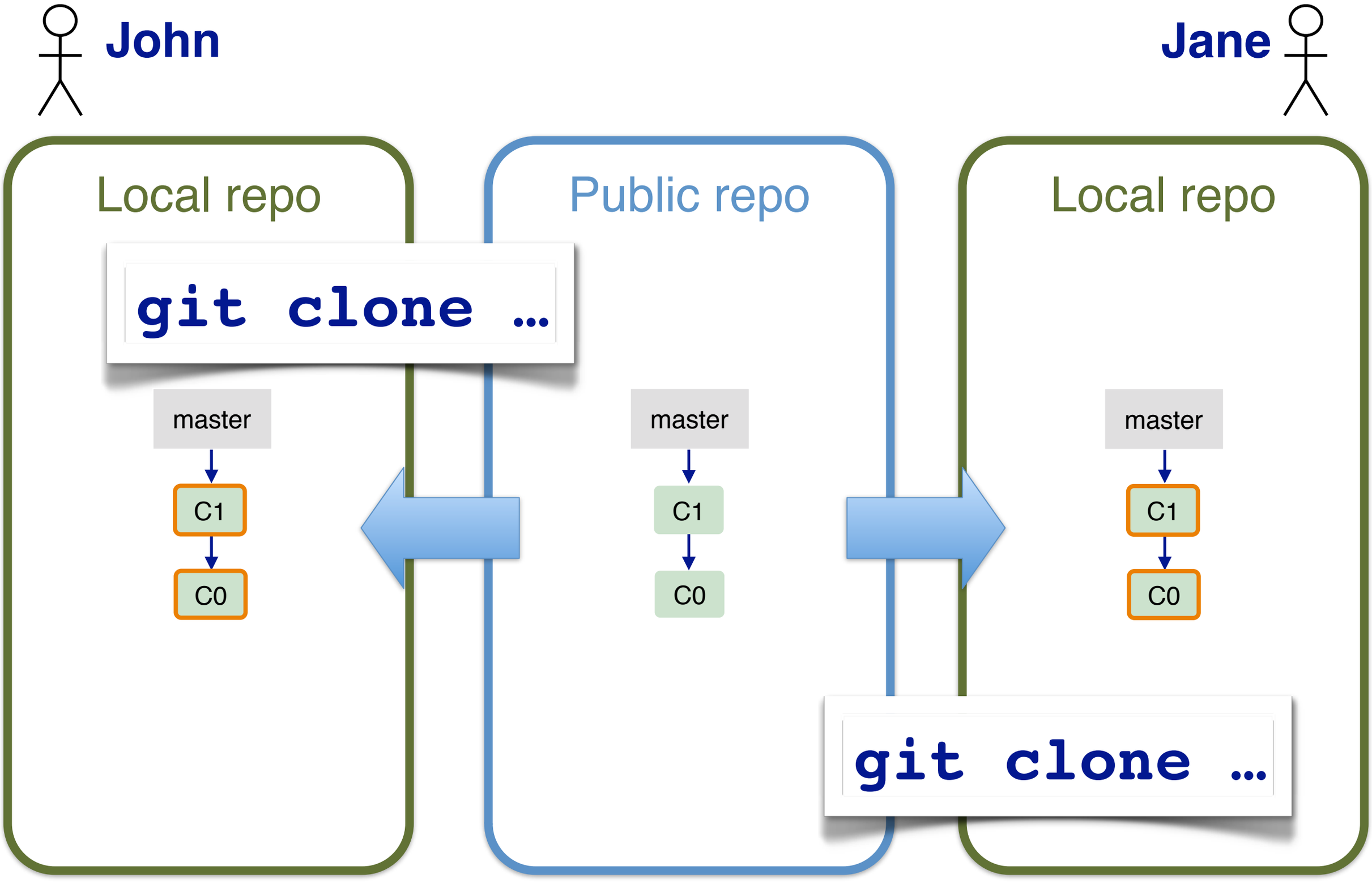
Jane

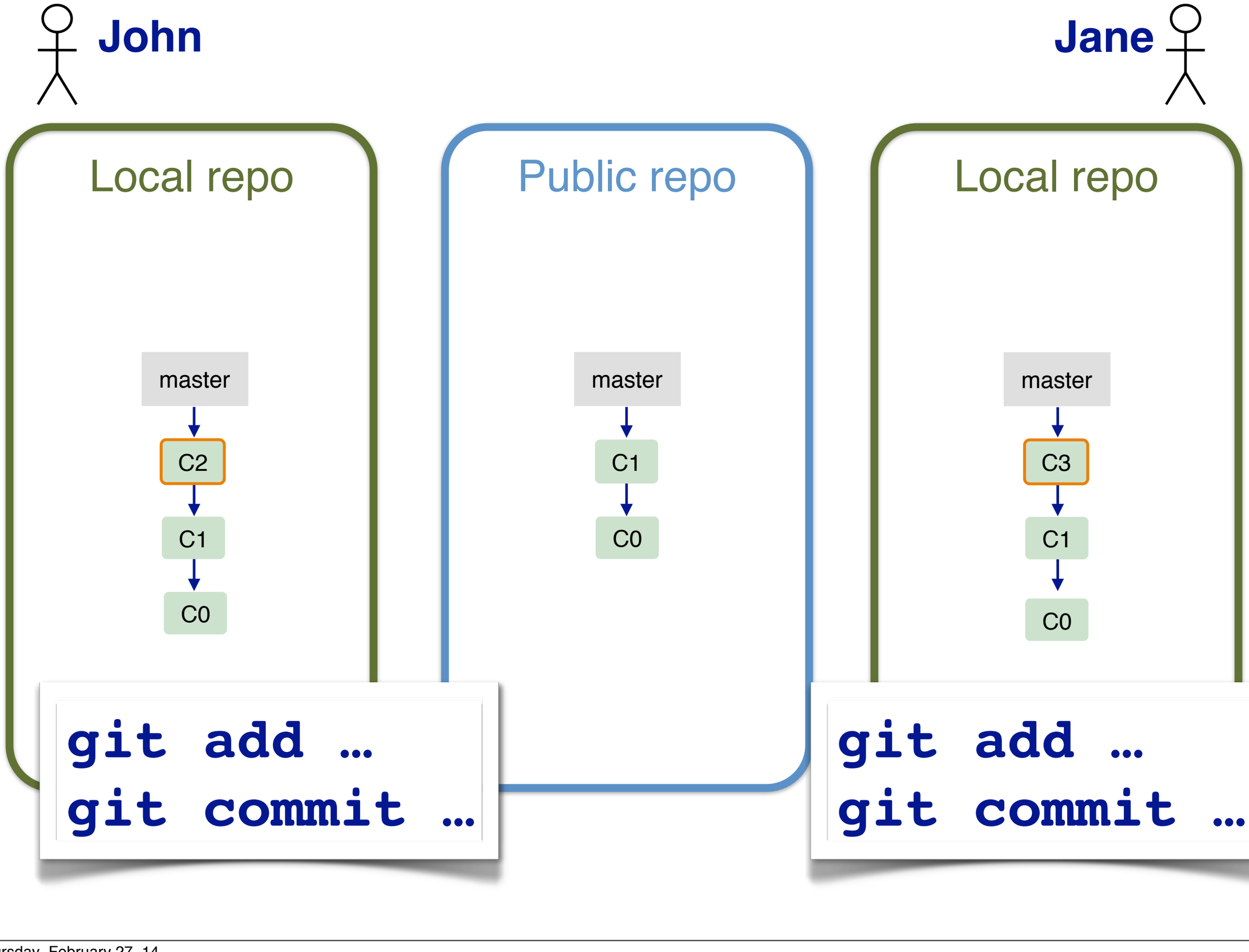
Local repo

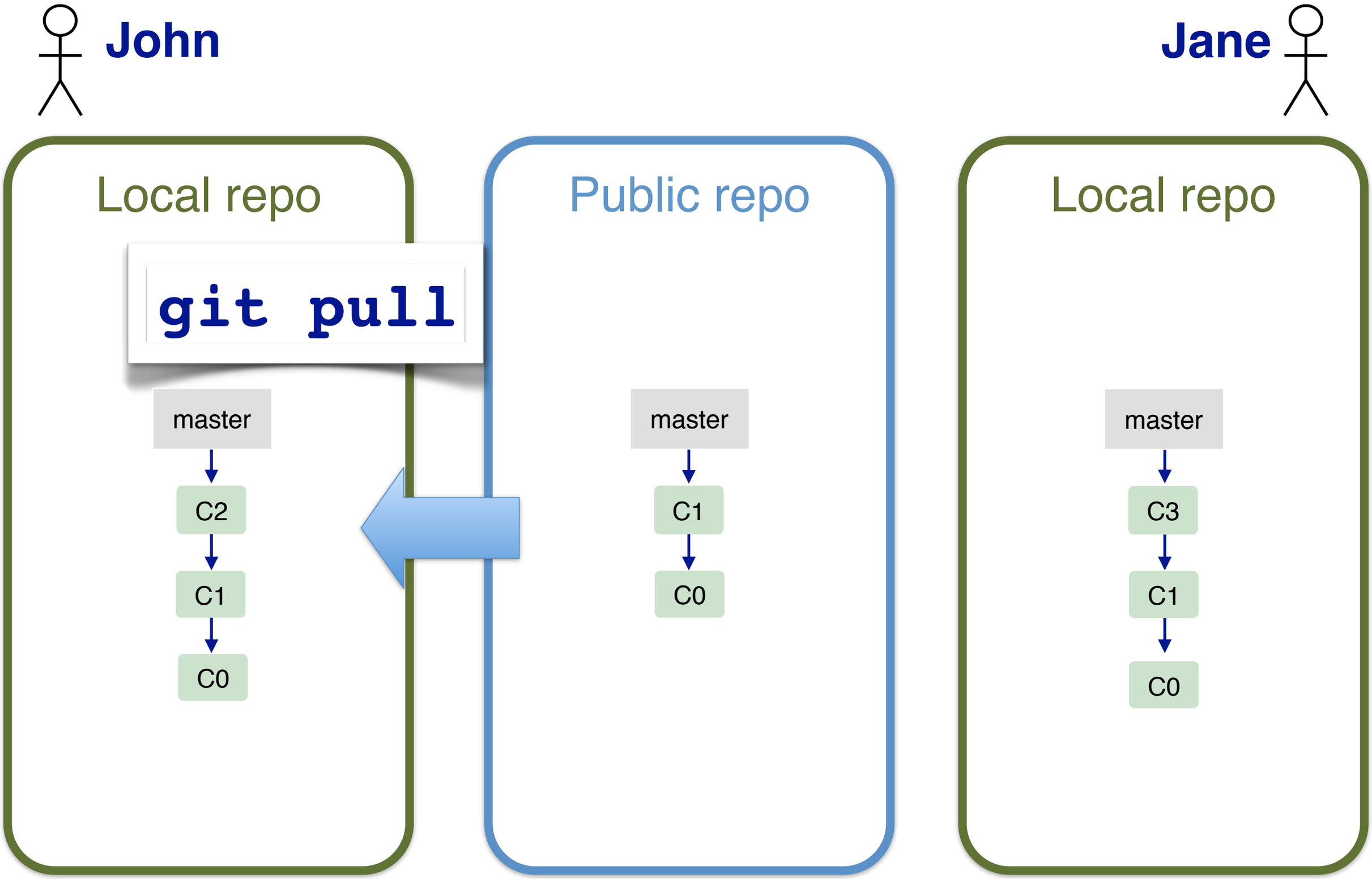
Public repo



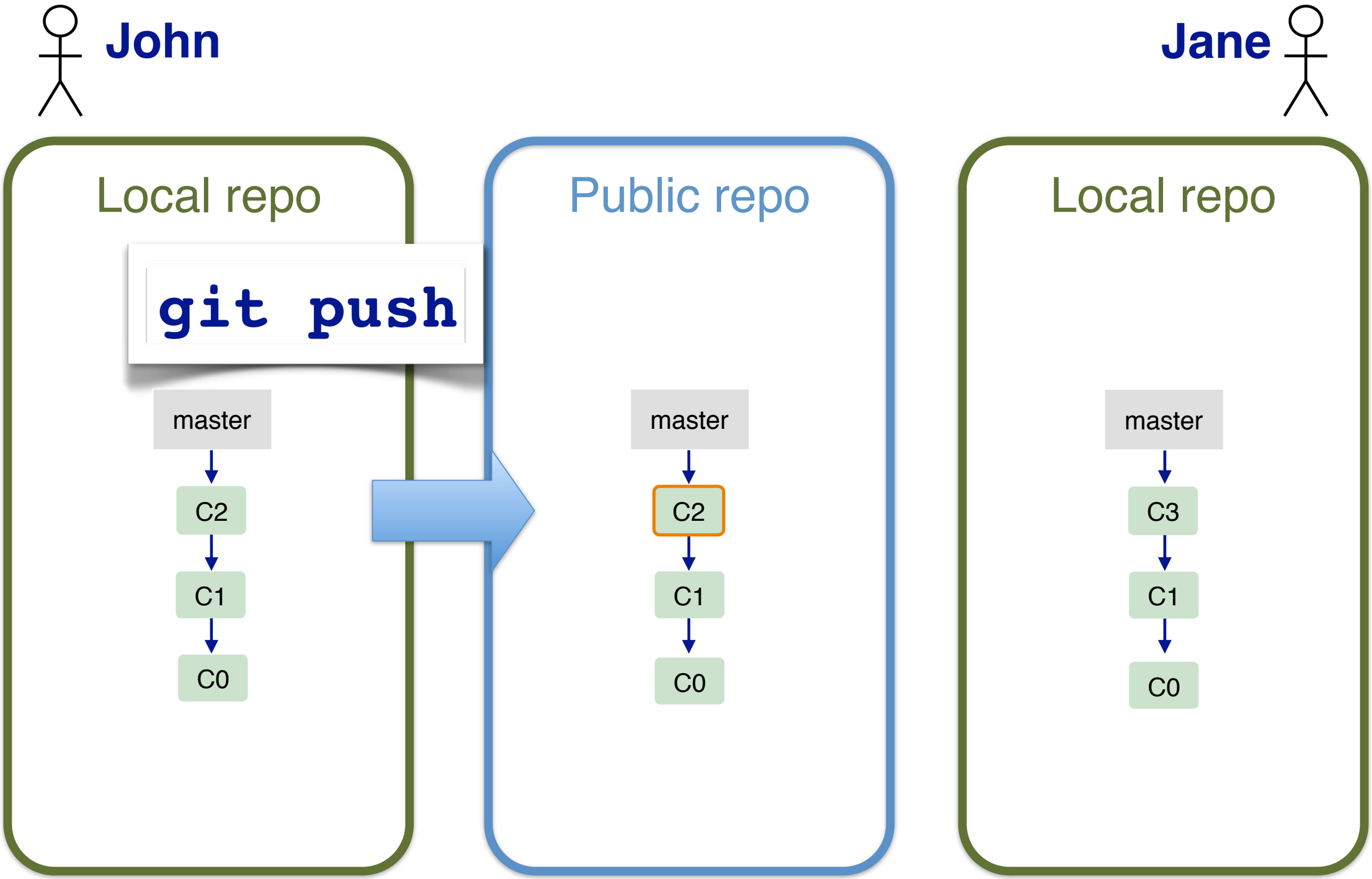
Local repo

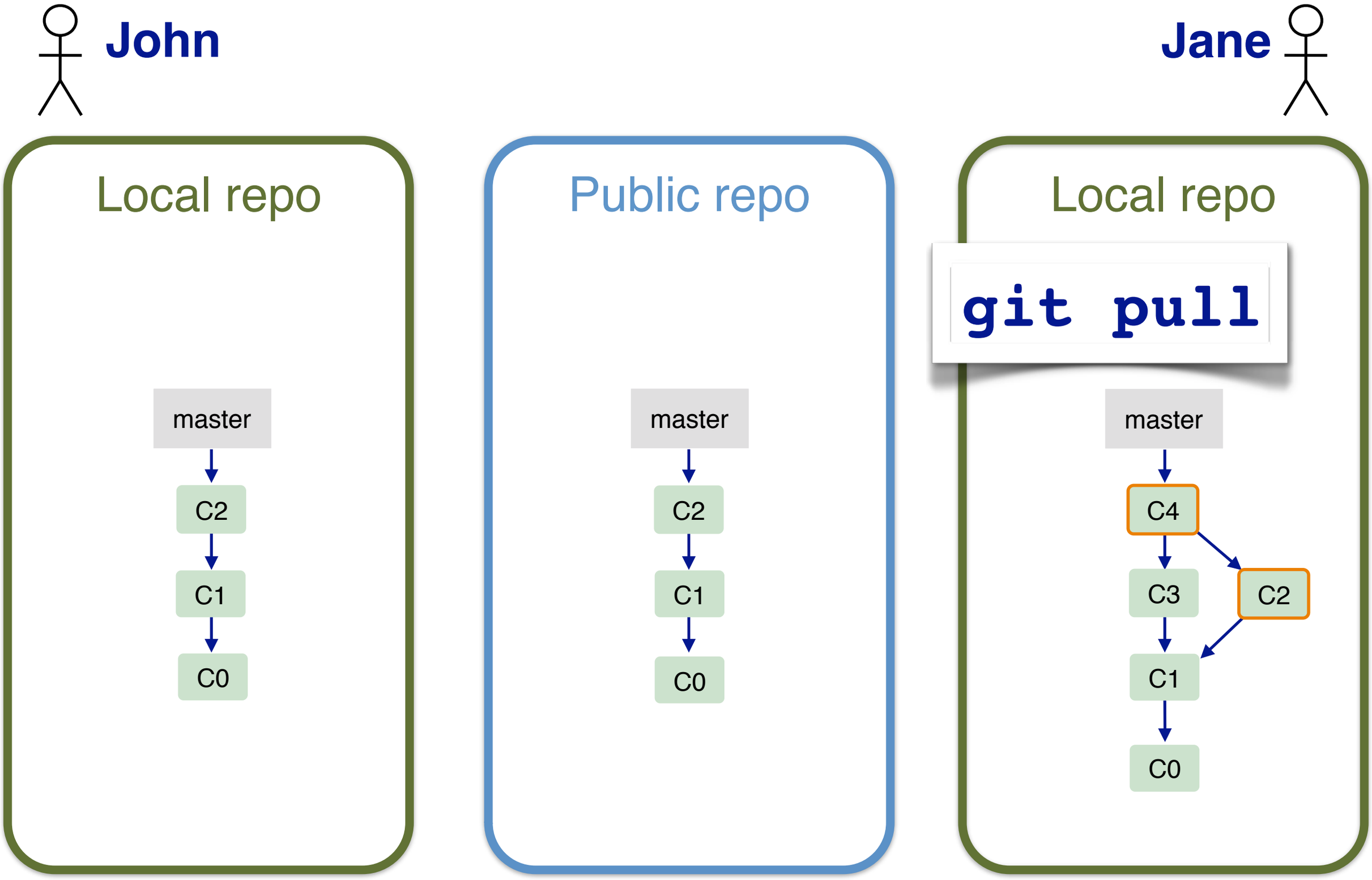

John
Jane
Local repo
Public repo
Local repo
git clone …
master
C1
C0
master
C1
C0
master
C1
C0
git clone …


John
Jane
Local repo
Public repo
Local repo
master
C2
C1
C0
master
C1
C0
master
C3
C1
C0
git add …
git commit …
git add …
git commit …

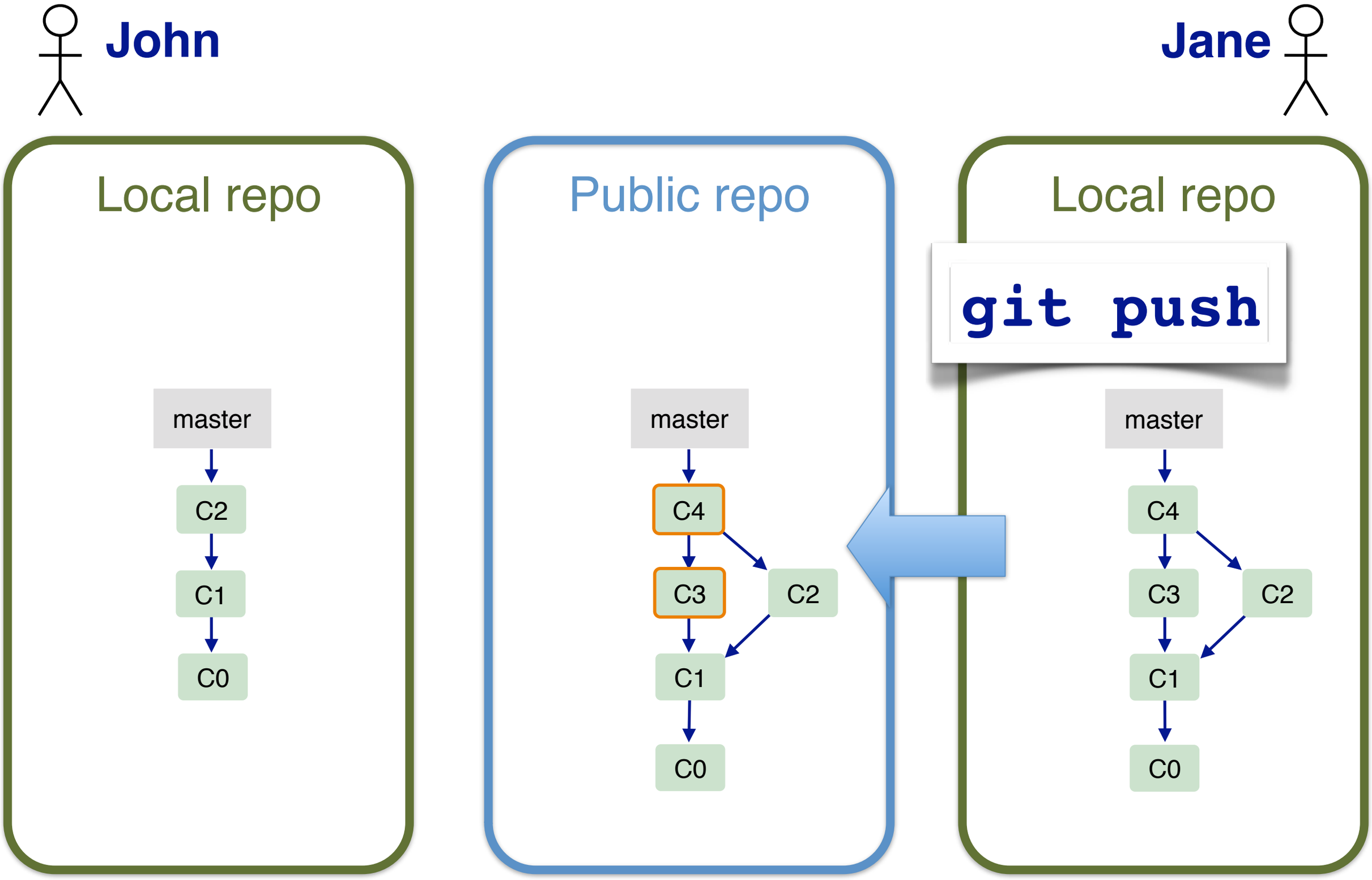


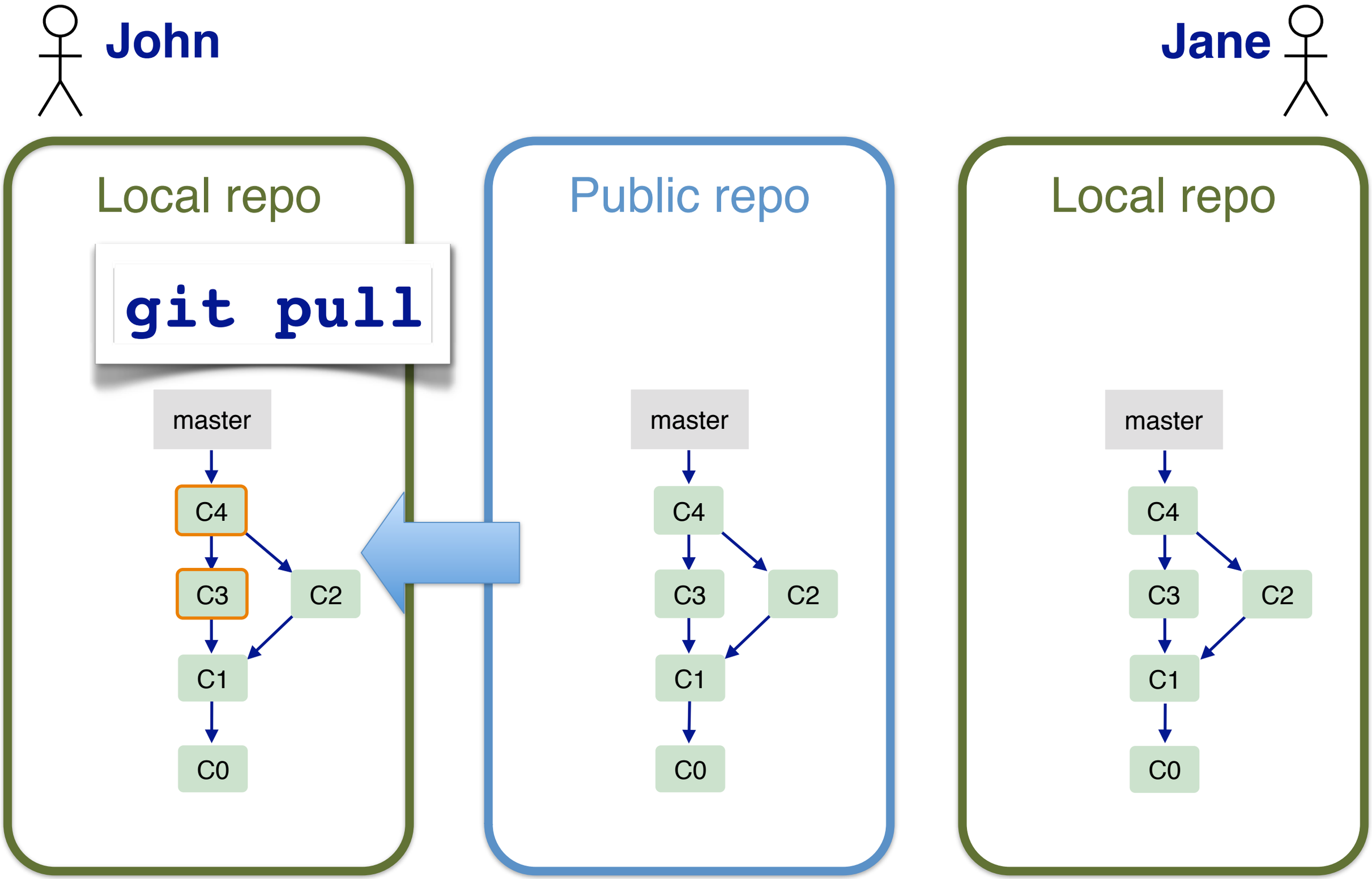
(nothing new to pull)


John
Jane
Local repo
Public repo
Local repo
git push
master
C2
C1
C0
master
C2
C1
C0
master
C3
C1
C0



NB: **`git pull`** = fetch + merge


John
Jane
Local repo
Public repo
Local repo
git push
master
C2
C1
C0
master
C4
C3
C2
C1
C0
master
C4
C3
C2
C1
C0


John
Jane
Local repo
Public repo
Local repo
git pull
master
C4
C3
C2
C1
C0
master
C4
C3
C2
C1
C0
master
C4
C3
C2
C1
C0

Branching and merging

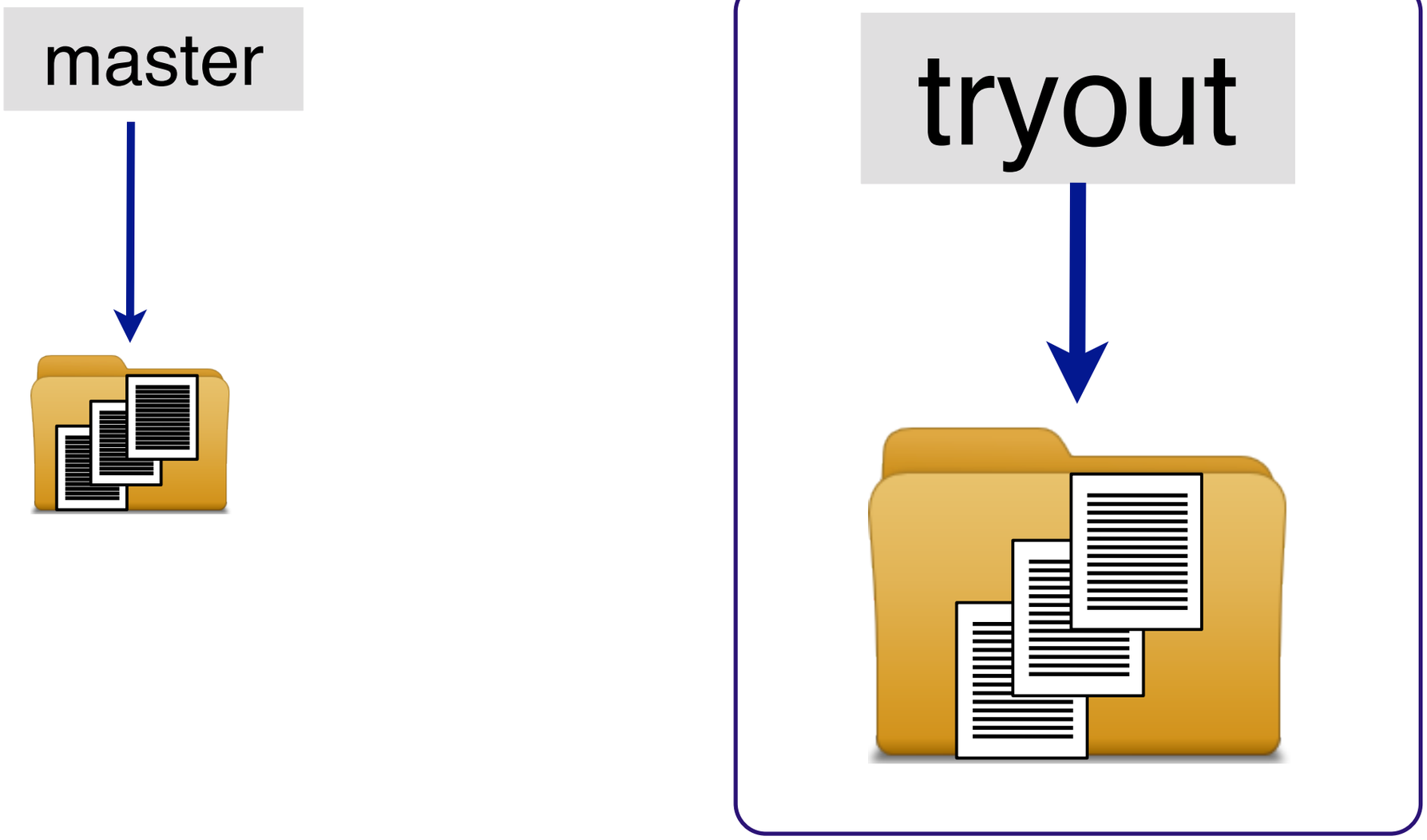
Branching is like “Save as...” on an entire directory

master

master

tryout

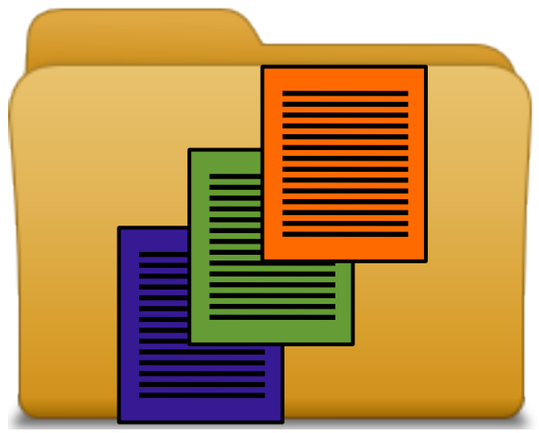
create branch



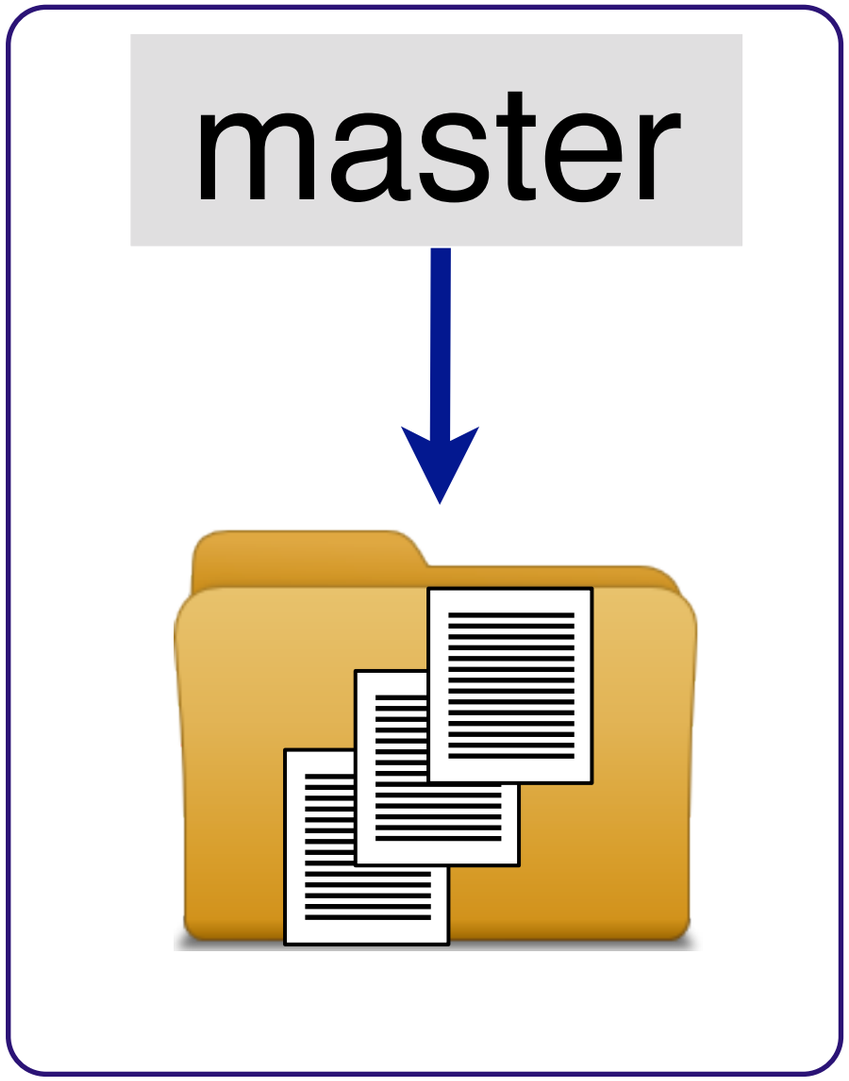
one branch
active at a time

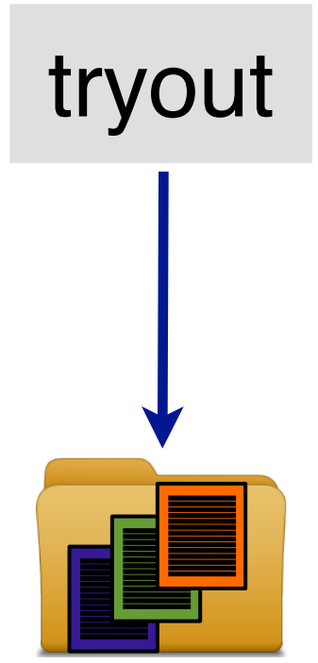
master

tryout

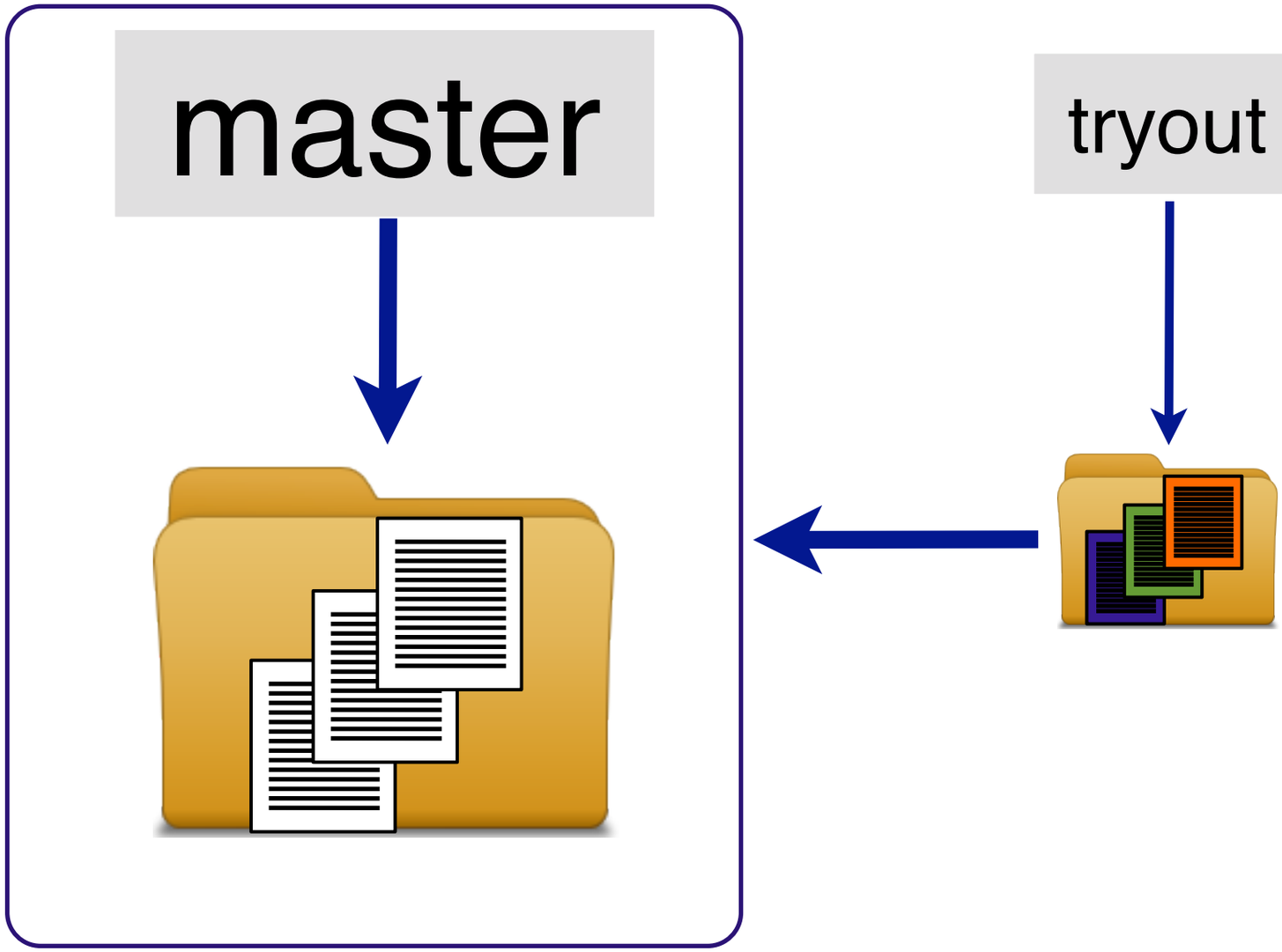


edit files in branch

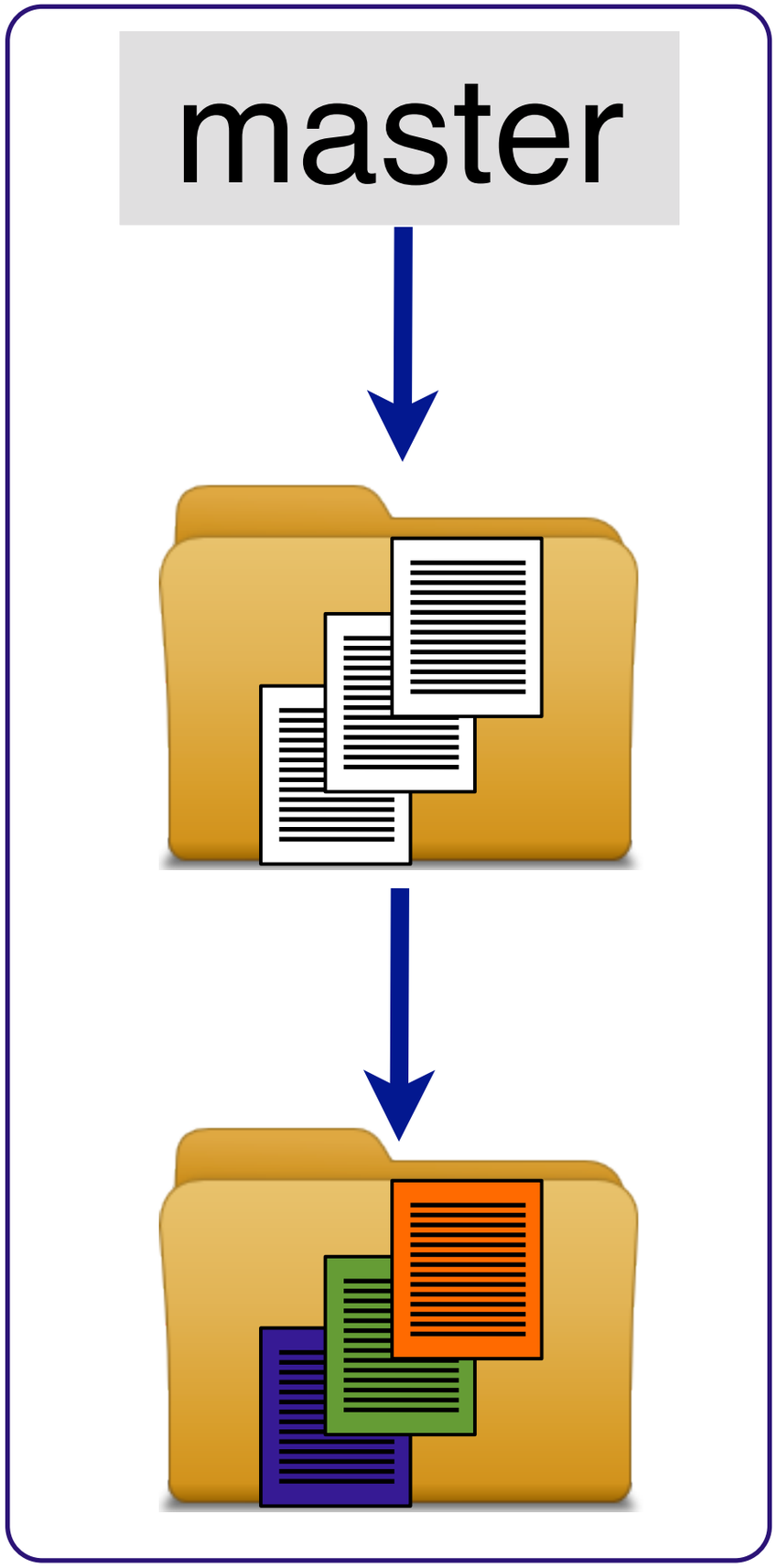




switch to master branch



merge with tryout branch



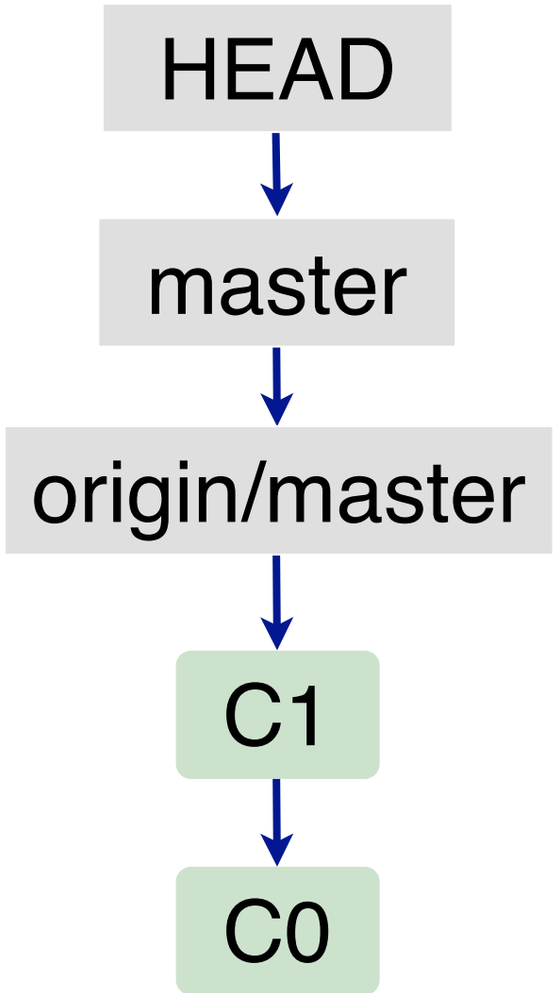
new version in master branch

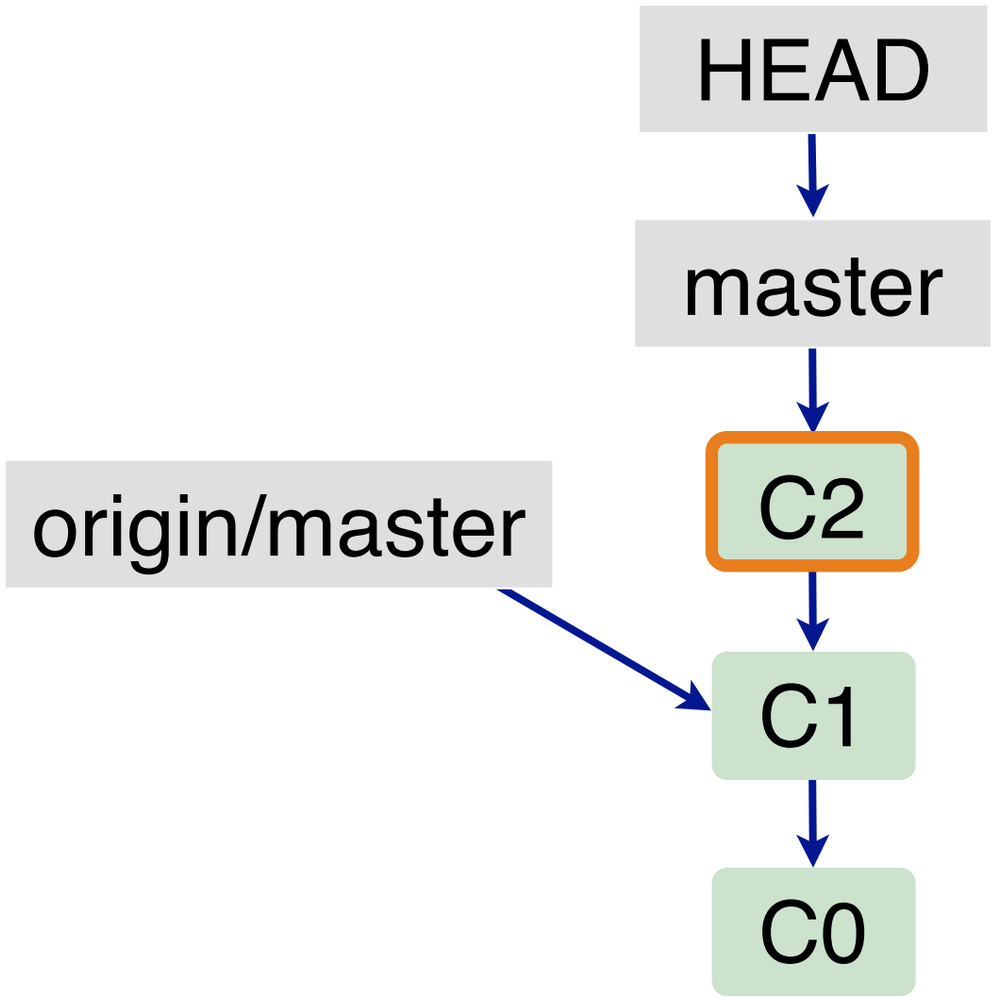
a branch
separates different
versions of your files

only one branch is active at a time

and now branching in git

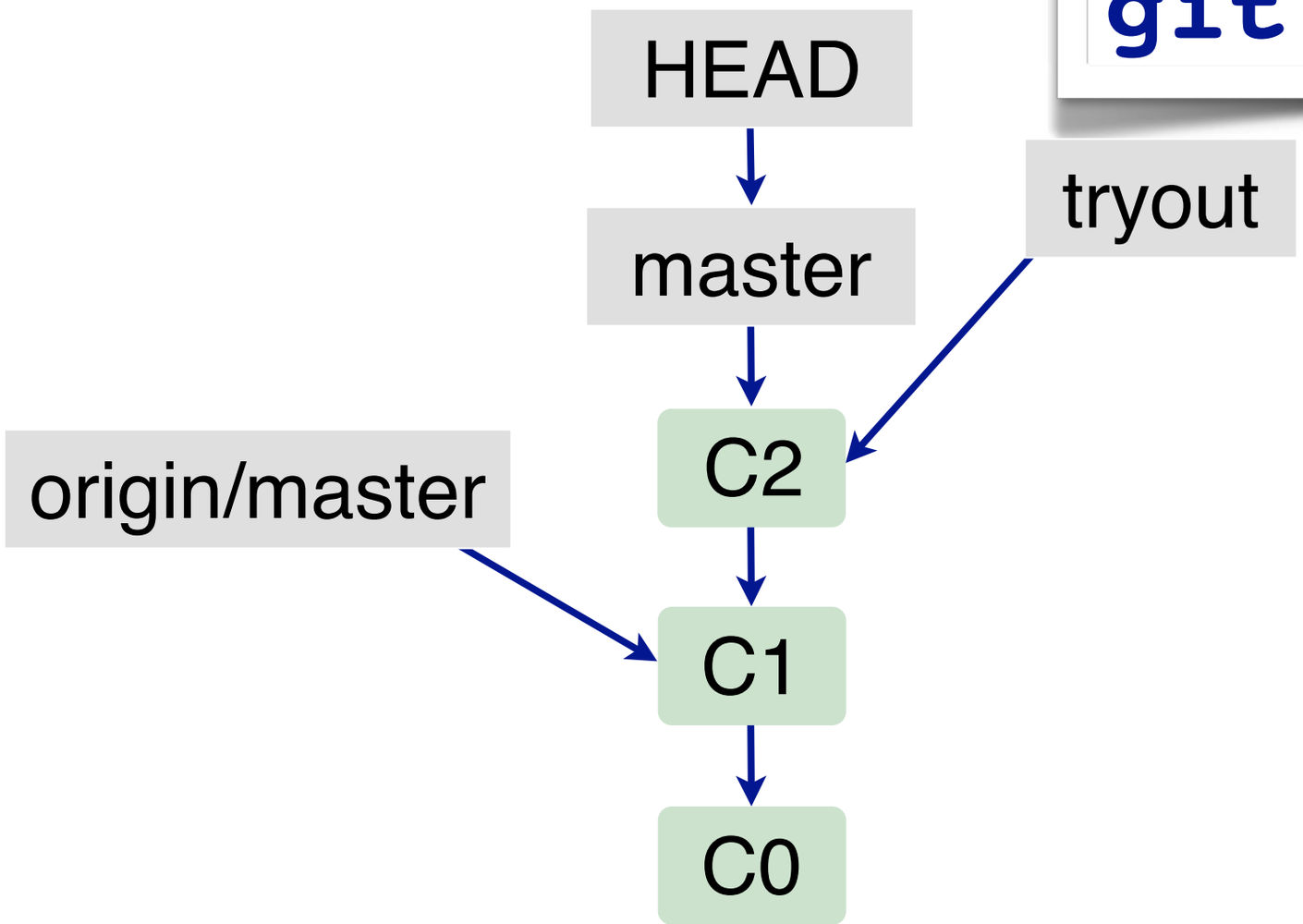
“origin” refers to the remote repo



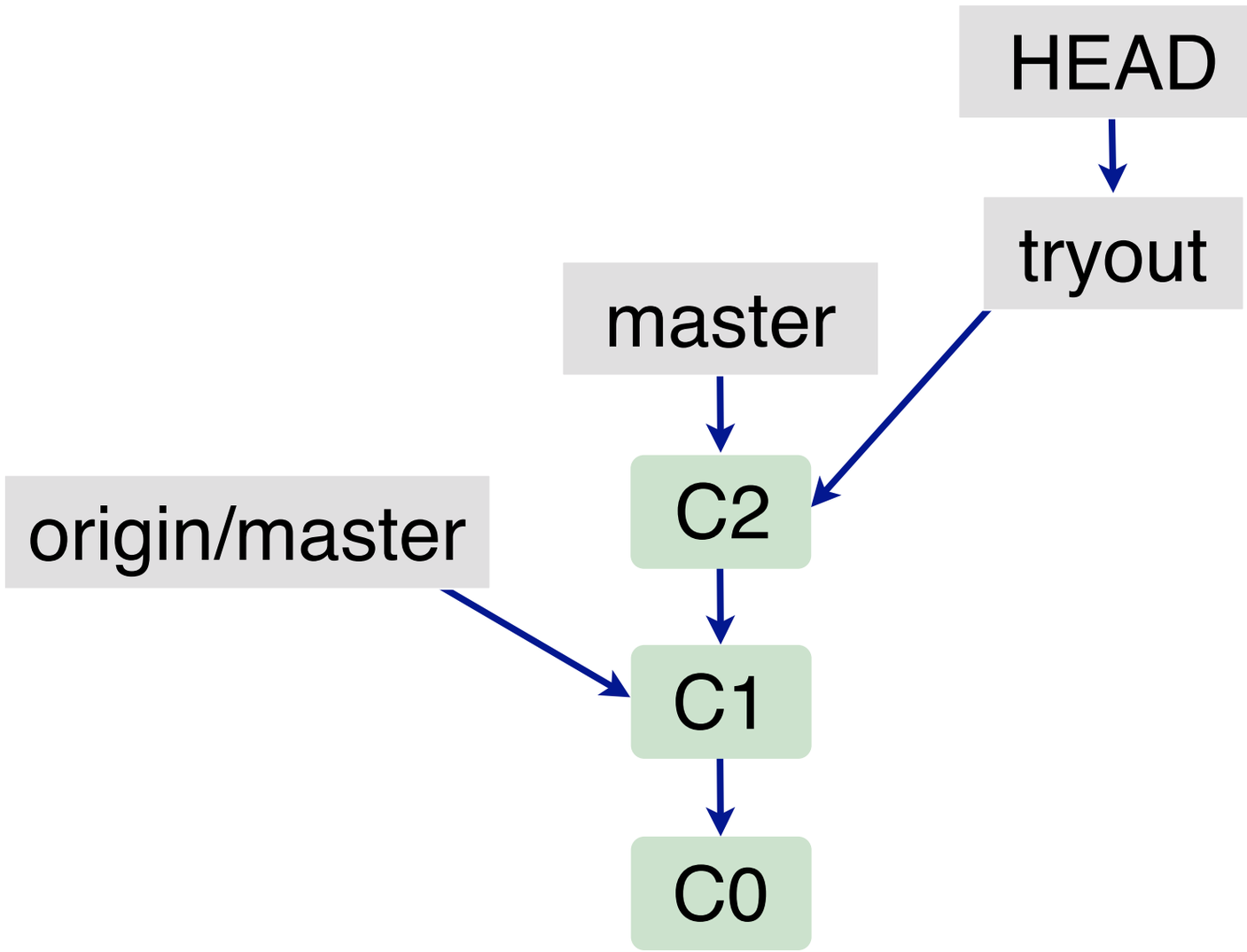


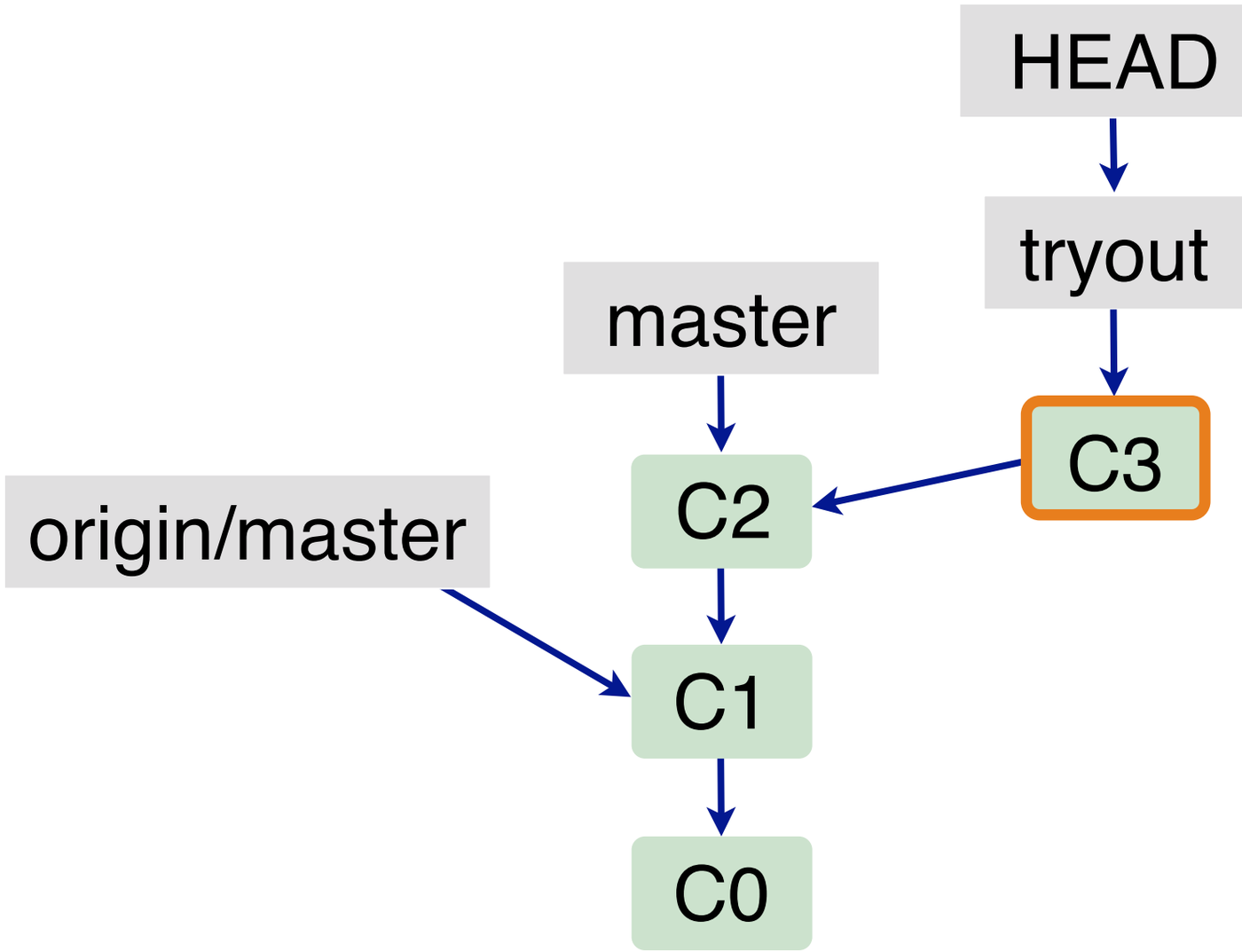

```
…
git commit …
```

```
git branch tryout
```

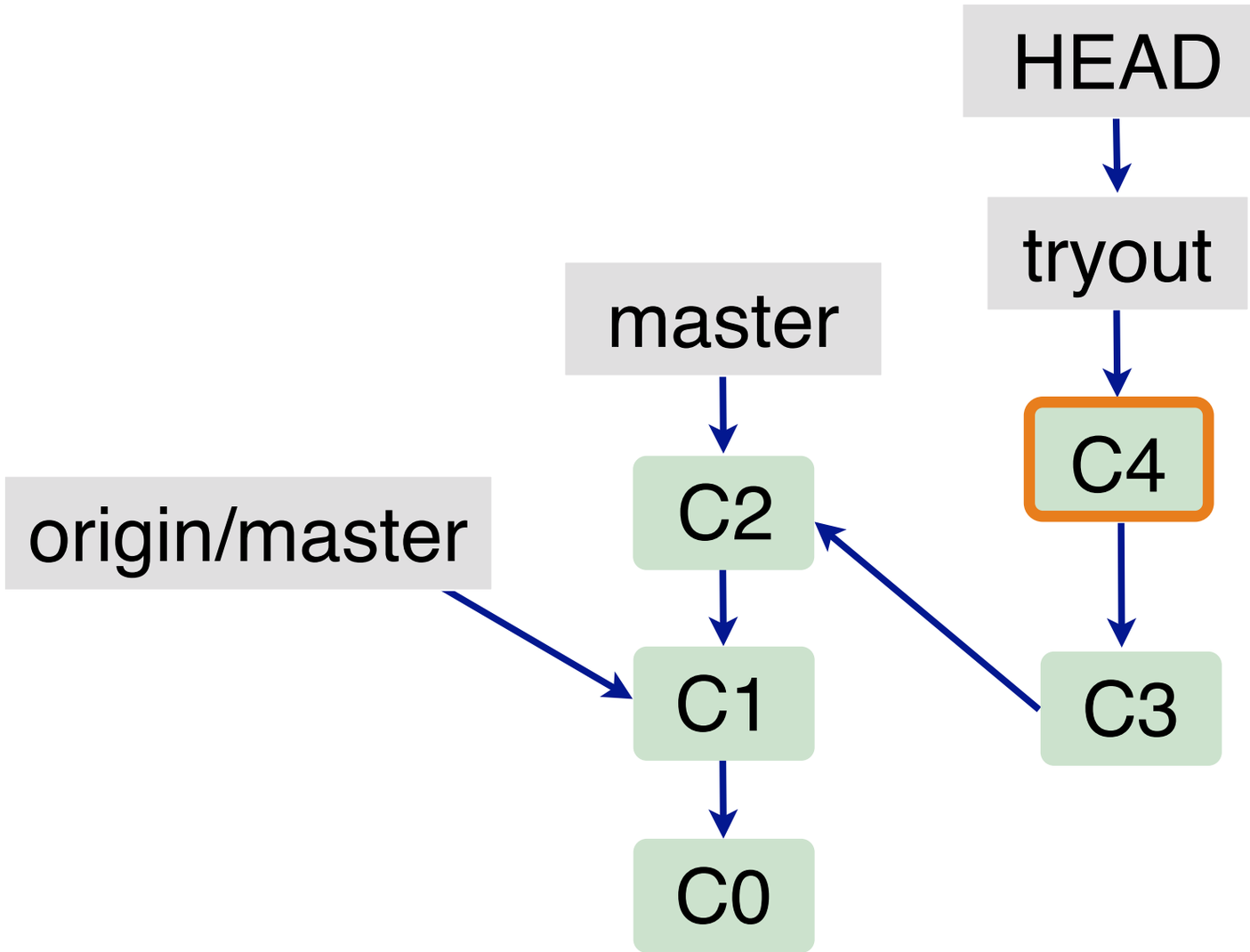


```
git checkout tryout
```



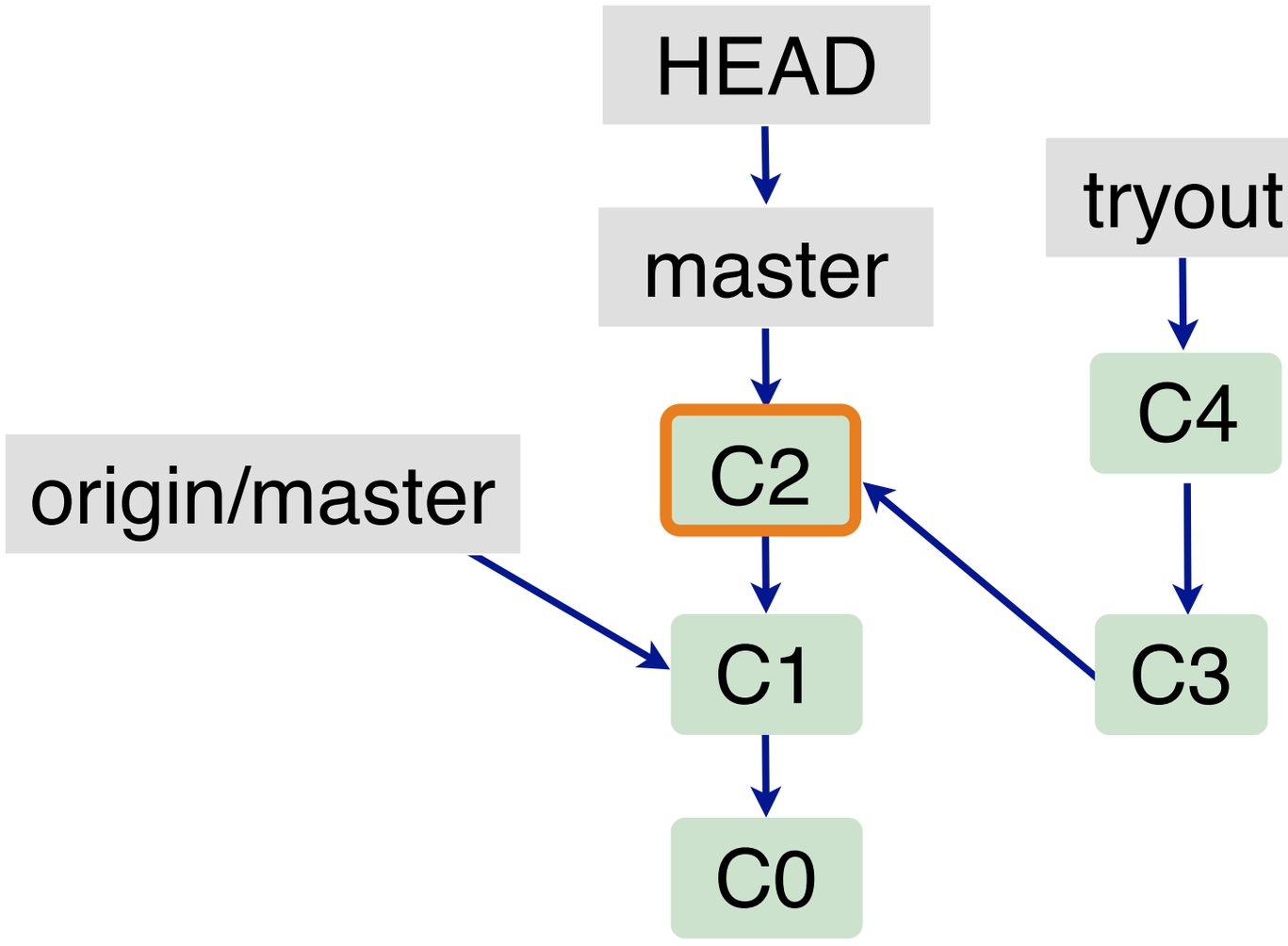


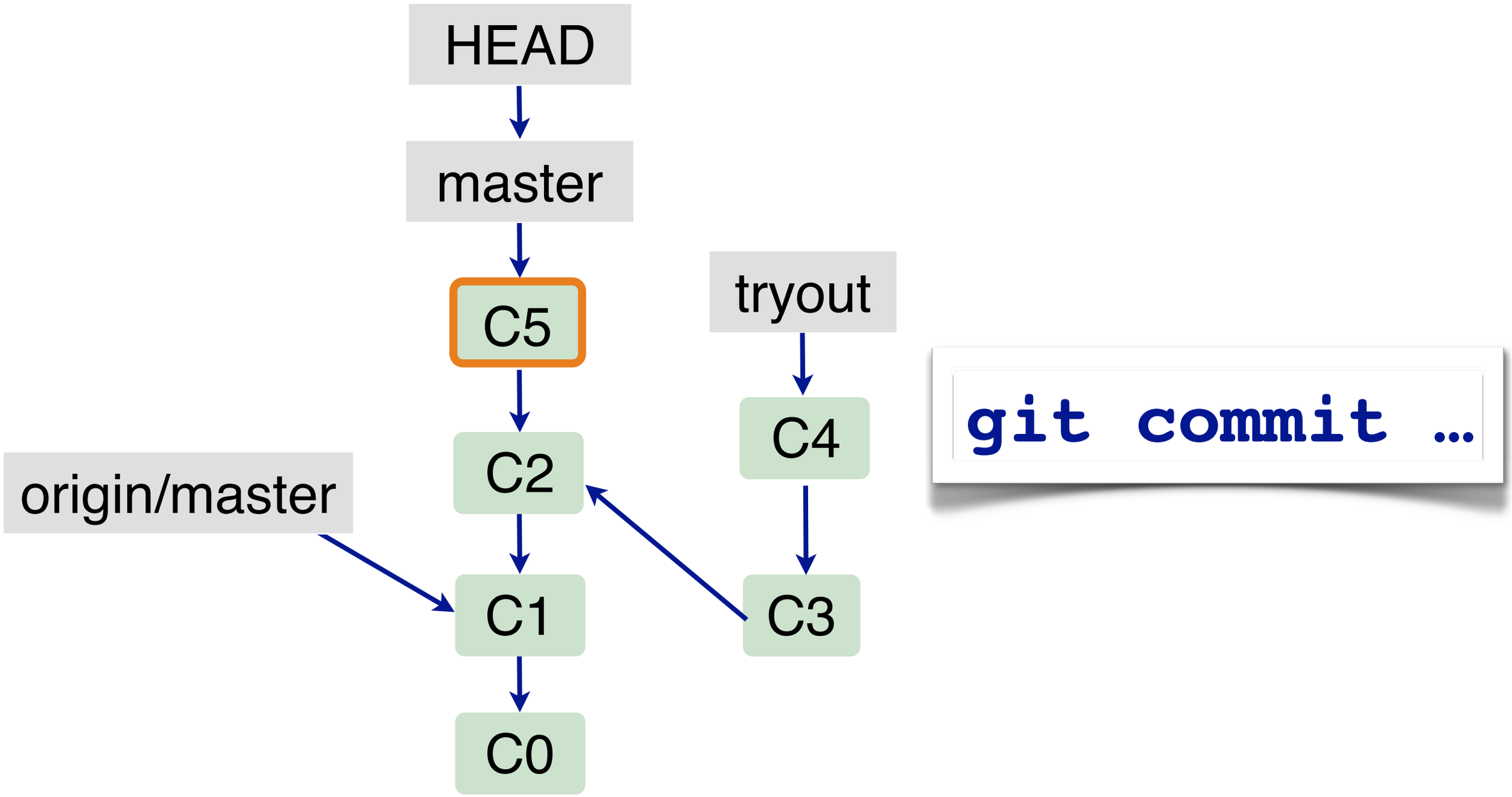

```
git commit …
```

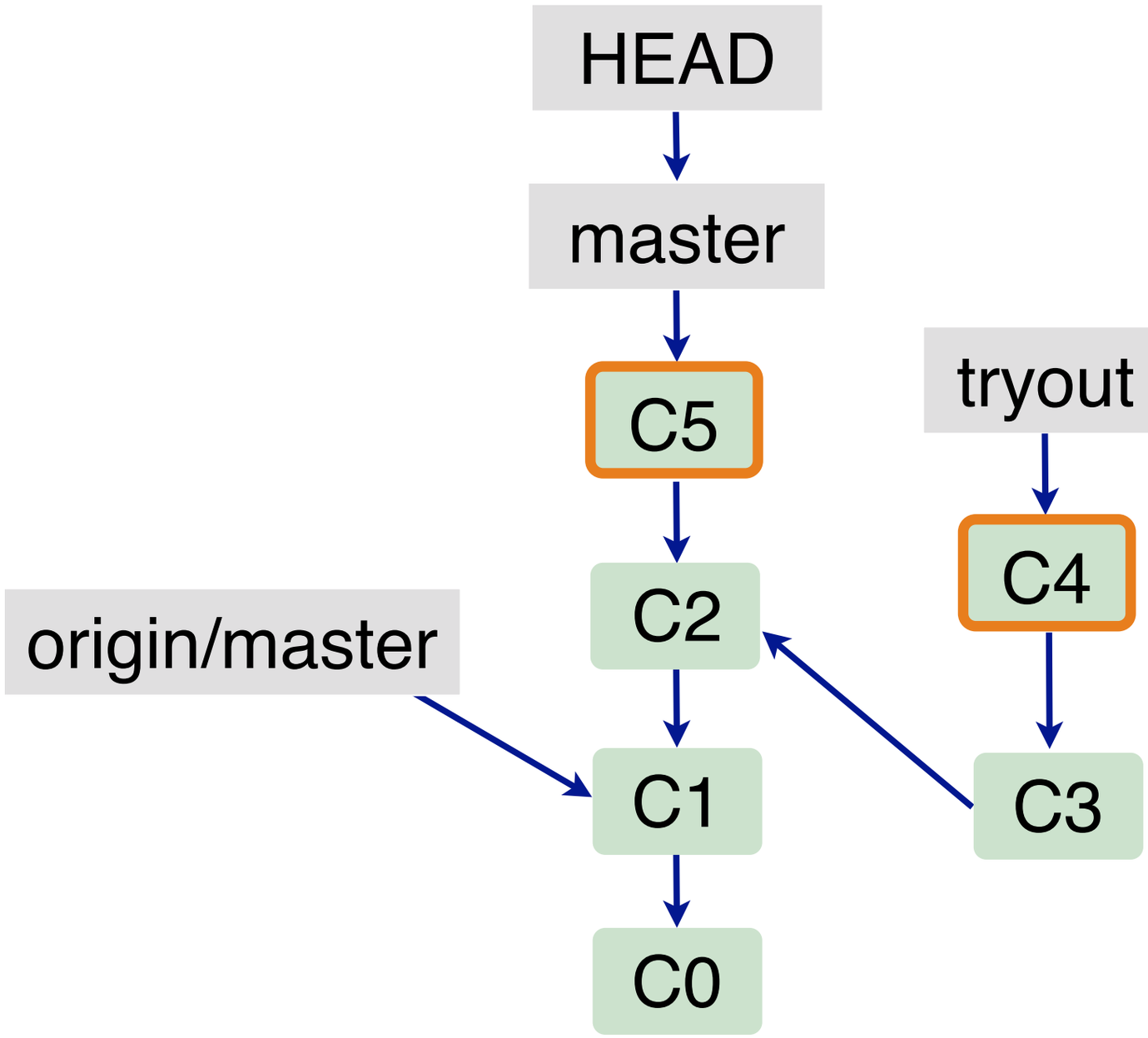



```
git commit …
```

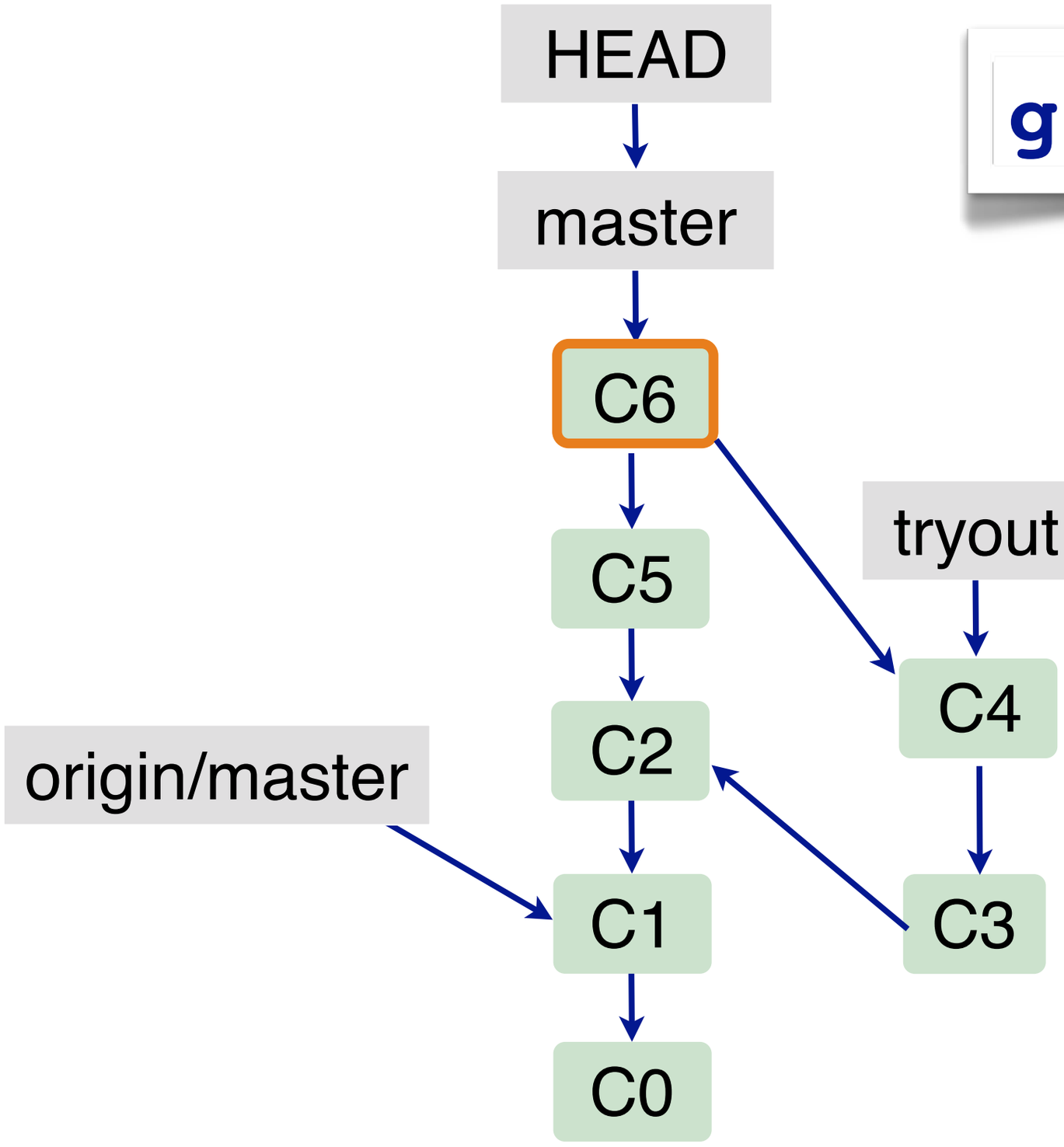
```
git checkout master
```

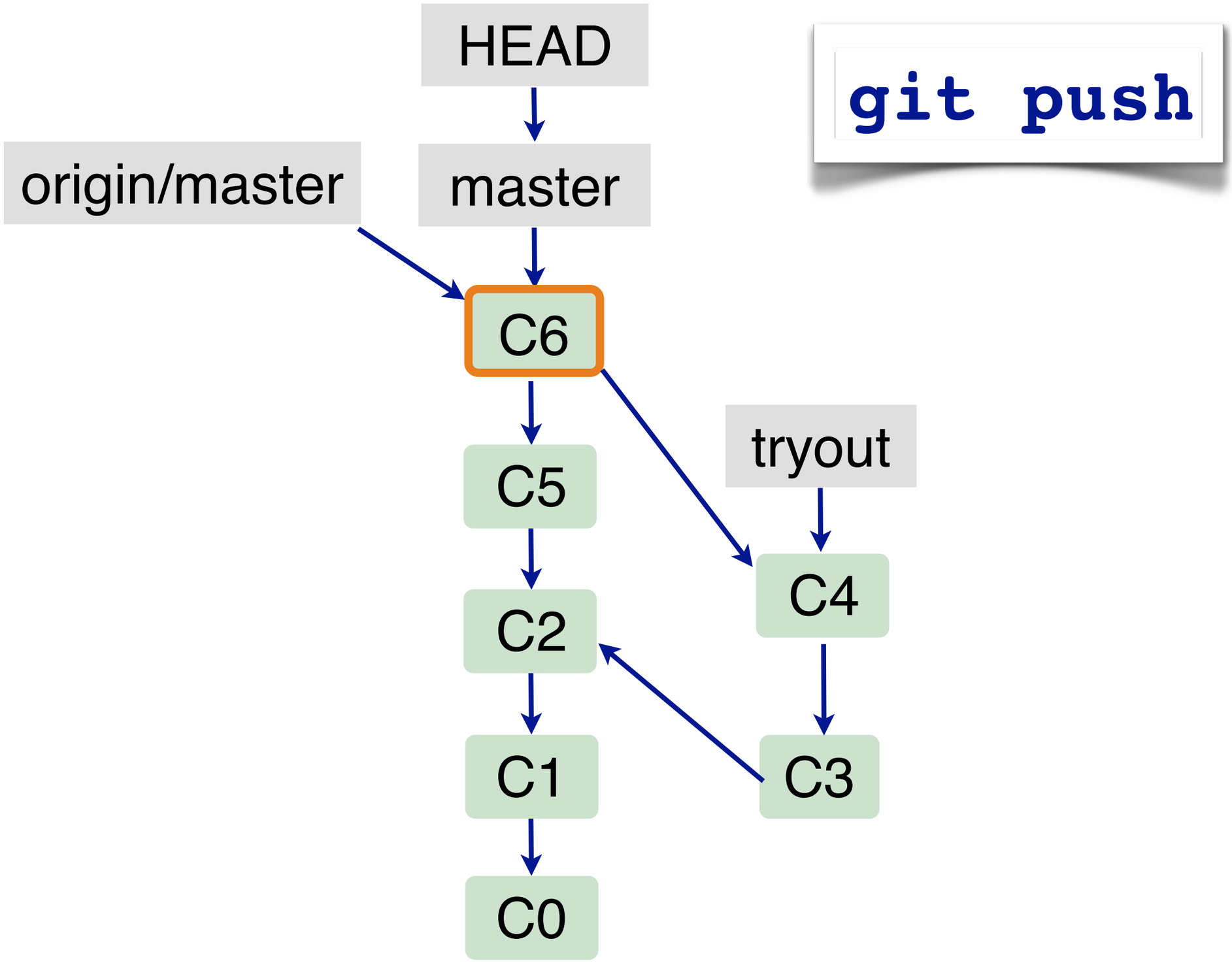



HEAD
master
C5
C2
C1
C0
origin/master
tryout
C4
C3
git commit …


HEAD
master
C5
C2
C1
C0
origin/master
tryout
C4
C3

```
git merge tryout
```






```
git push
```

More to git

More to git …

- Merging and mergetool
- Squashing commits when merging
- Resolving conflicts
- User authentication with ssh
- gitx and other graphical tools
- git configure — remembering your name
- git remote — multiple remote repos
- github — an open source public repo
- …

Resources

http://git-scm.com/

http://book.git-scm.com/index.html

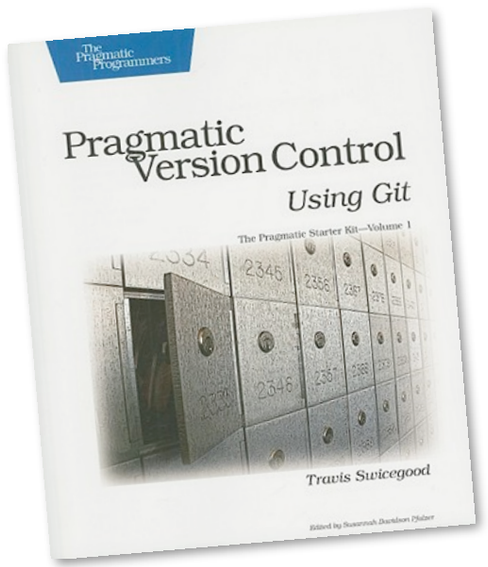
Getting Git

Scott Chacon

http://www.slideshare.net/chacon/getting-git

github
SOCIAL CODING

https://github.com/



http://oreilly.com/