

Distributed version control with git — a brief introduction

Andrei Chis

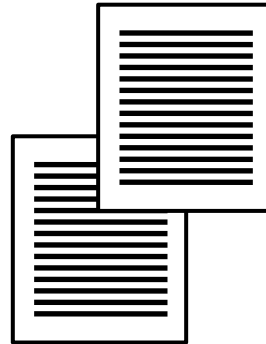
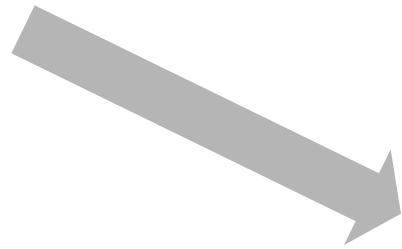
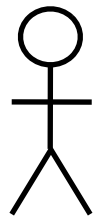
based on slides by
Oscar Nierstrasz



Why version control?

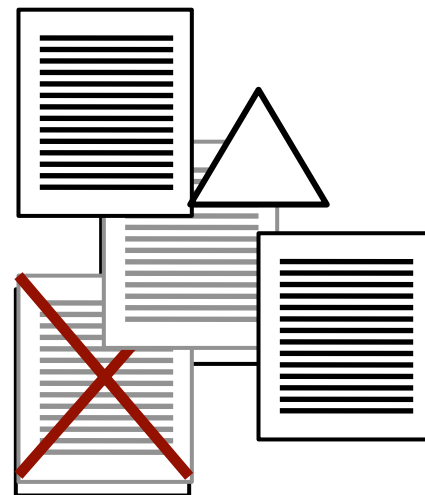
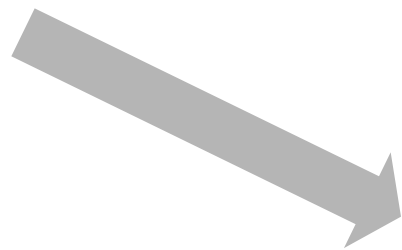
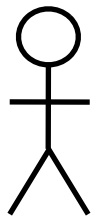
Why version control?

Bob

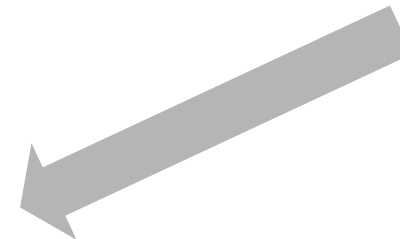
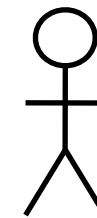


Why version control?

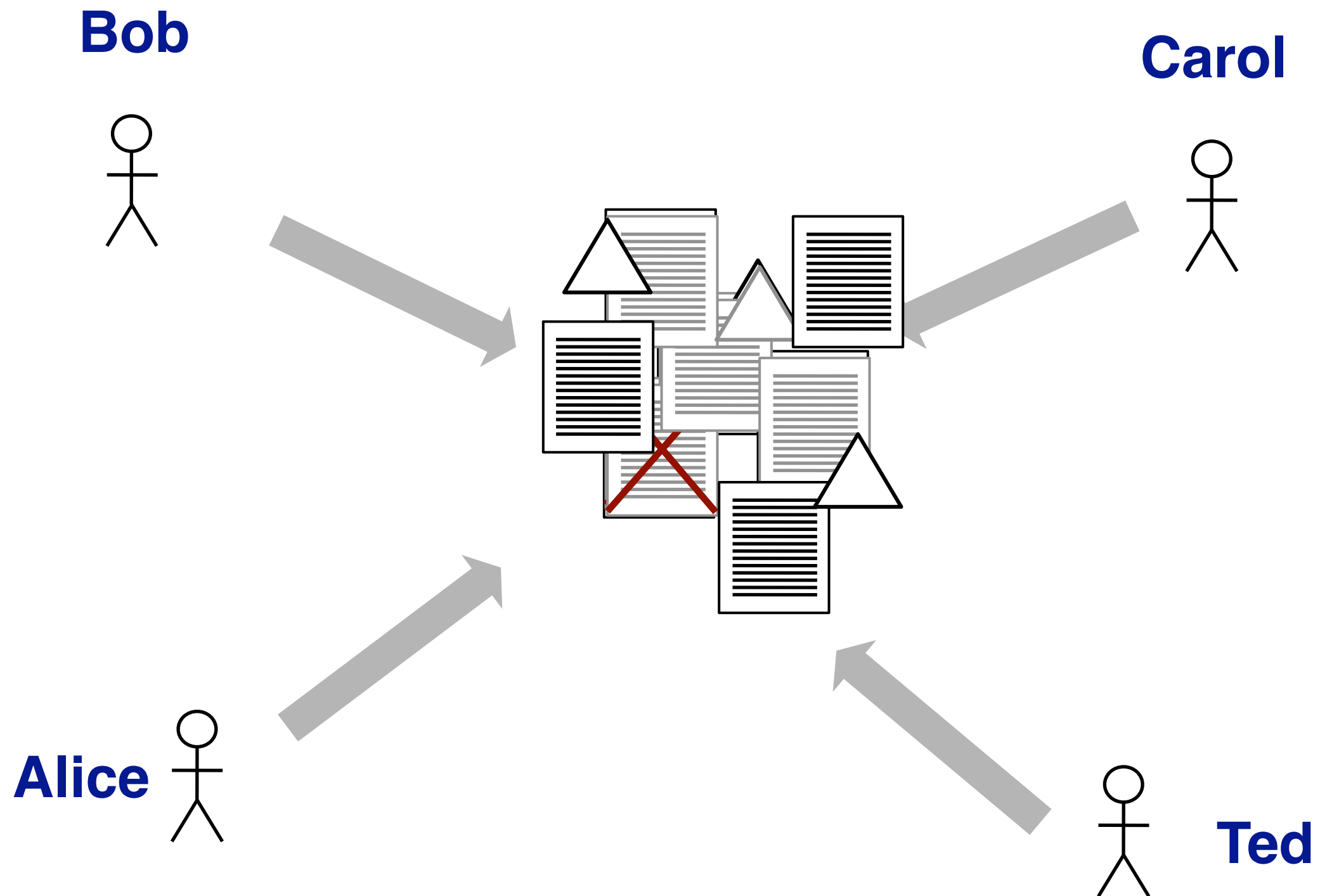
Bob



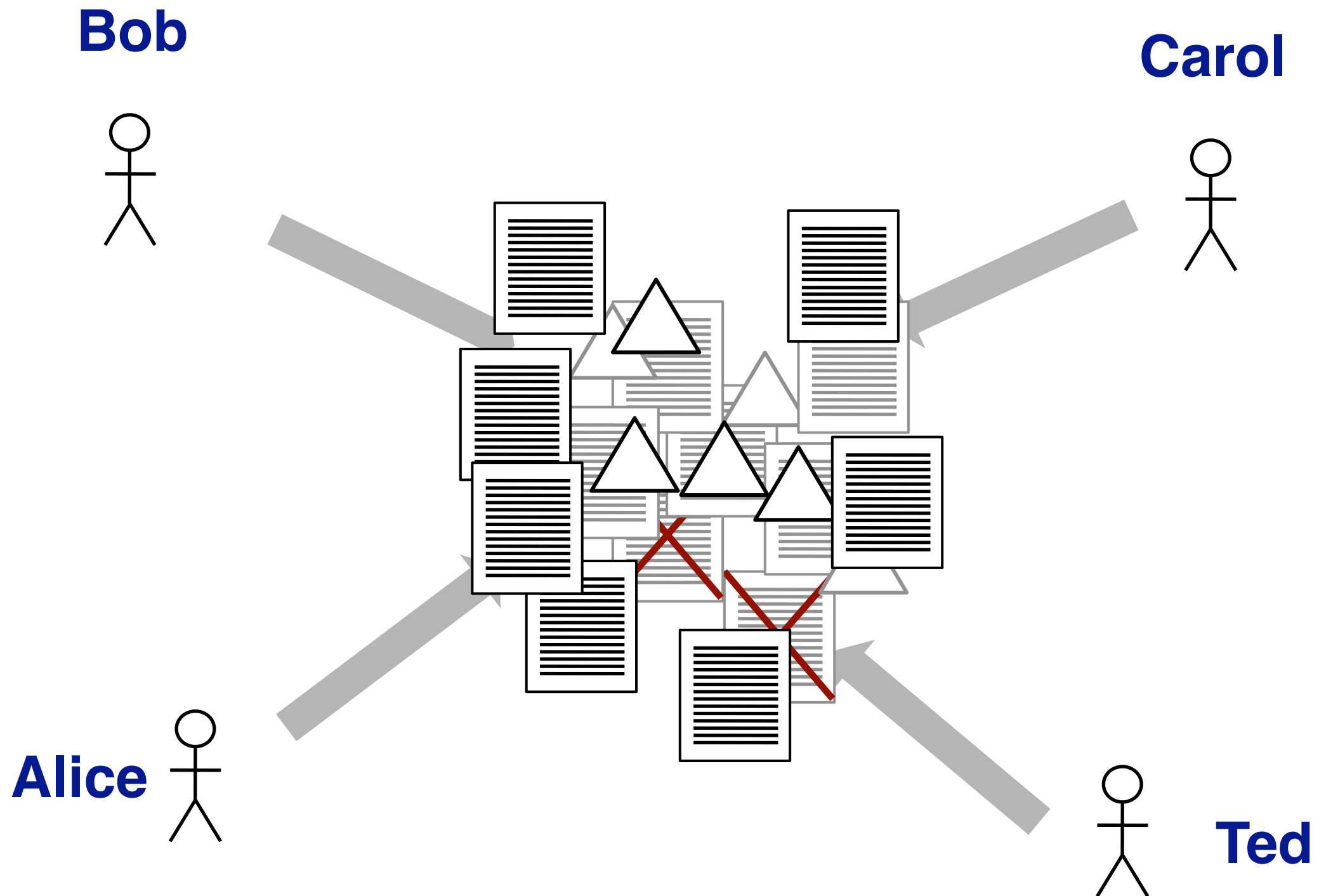
Carol



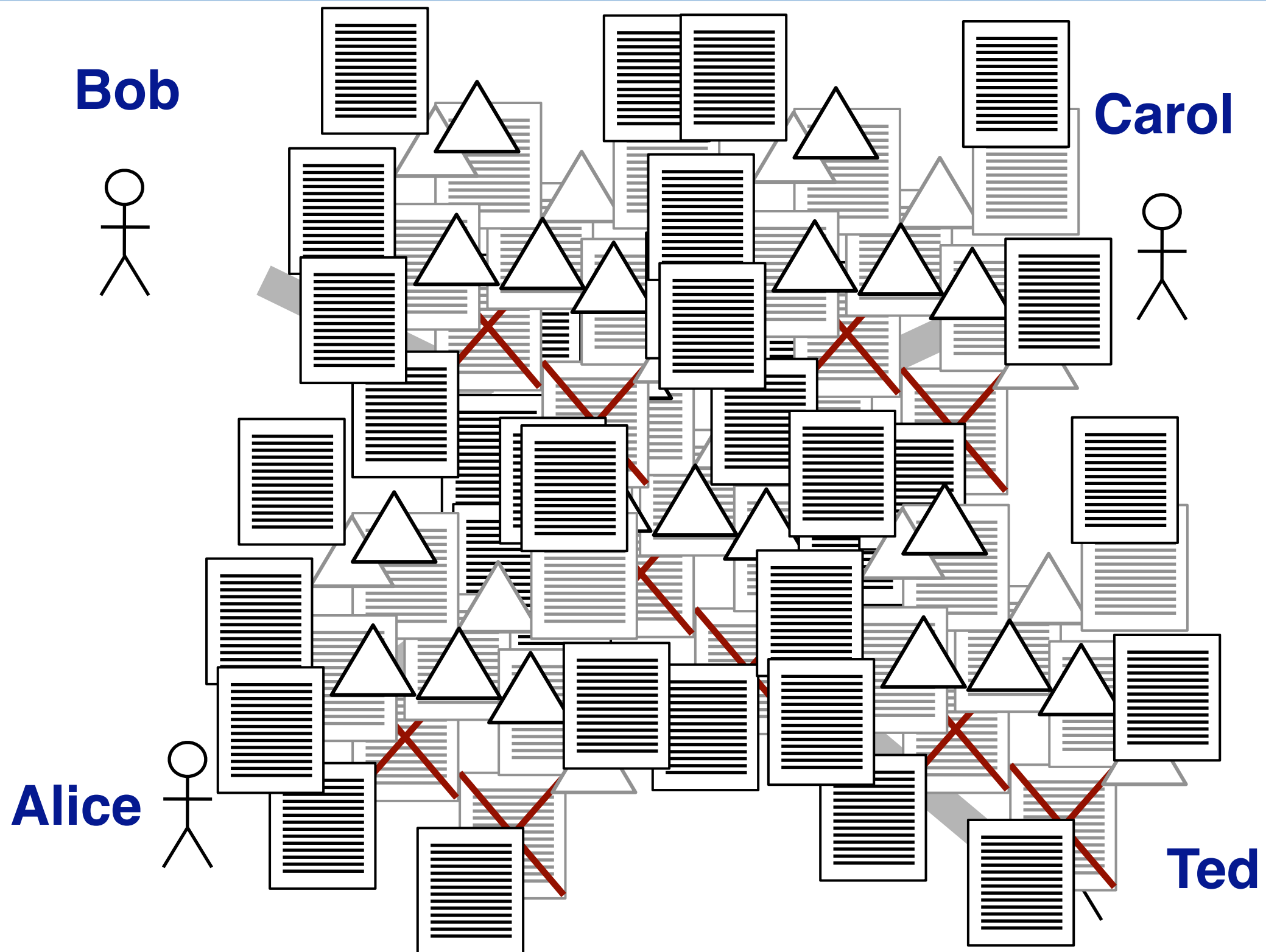
Why version control?



Why version control?



Why version control?



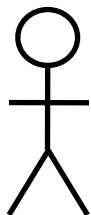
Why version control?



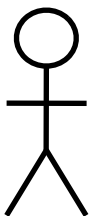
Why version control?

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

Bob

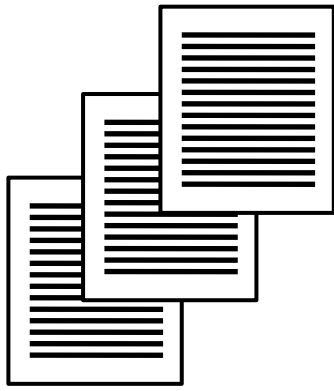
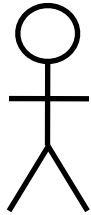


Carol

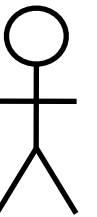


Repository

Bob



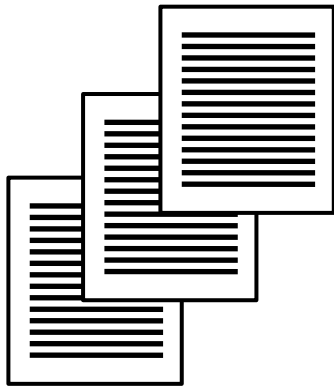
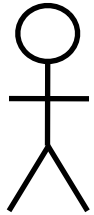
Carol



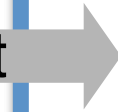
Repository



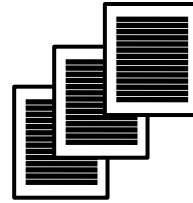
Bob



Snapshot

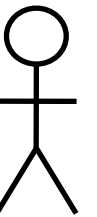


Version 1

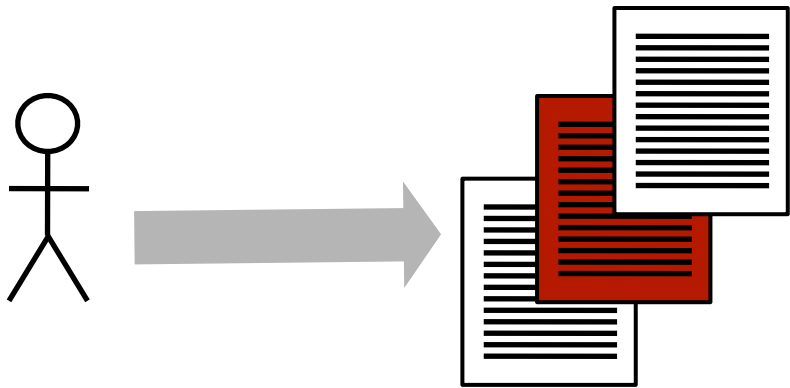


Repository

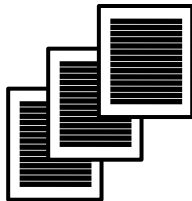
Carol



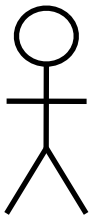
Bob



Version 1

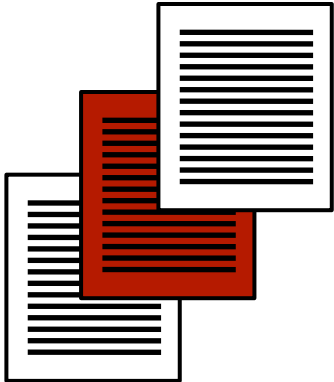
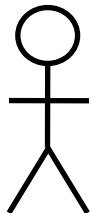


Carol

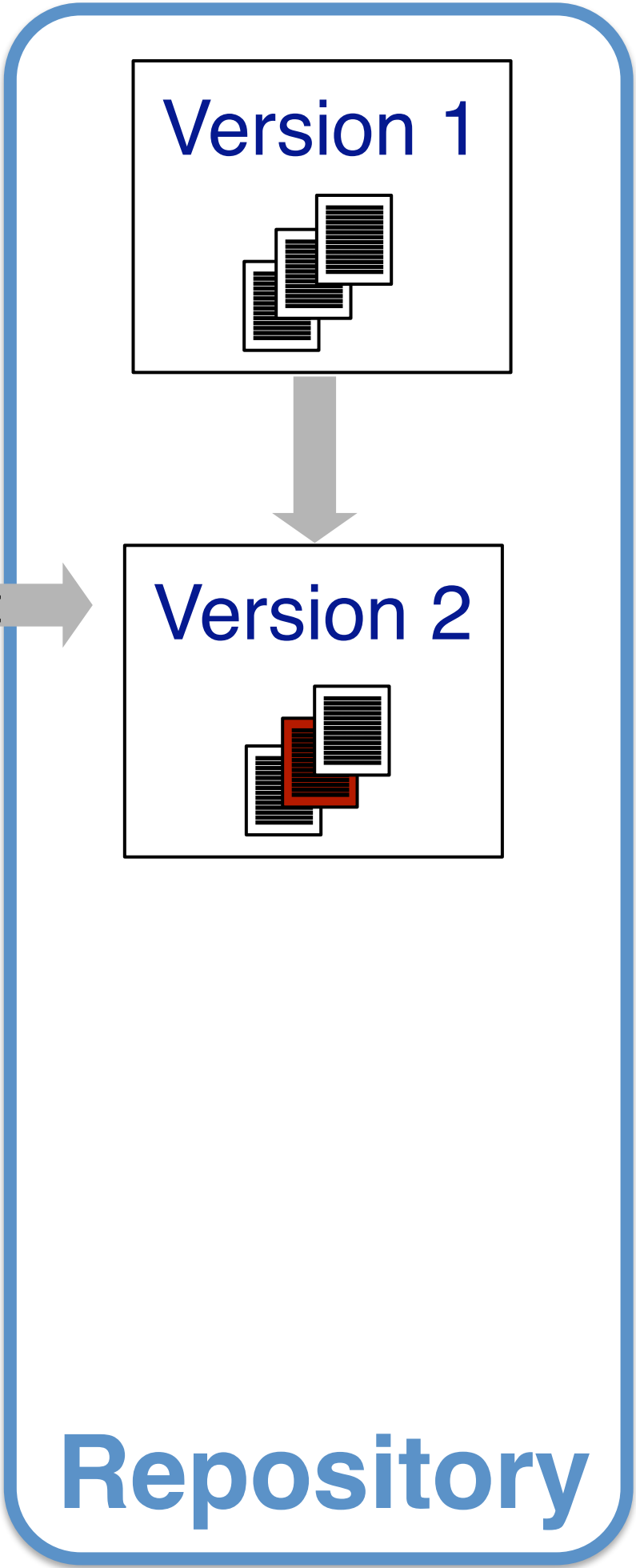


Repository

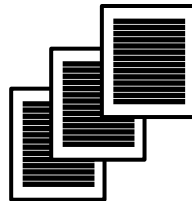
Bob



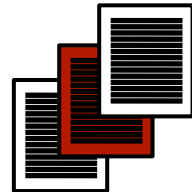
Snapshot



Version 1

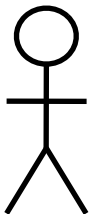


Version 2

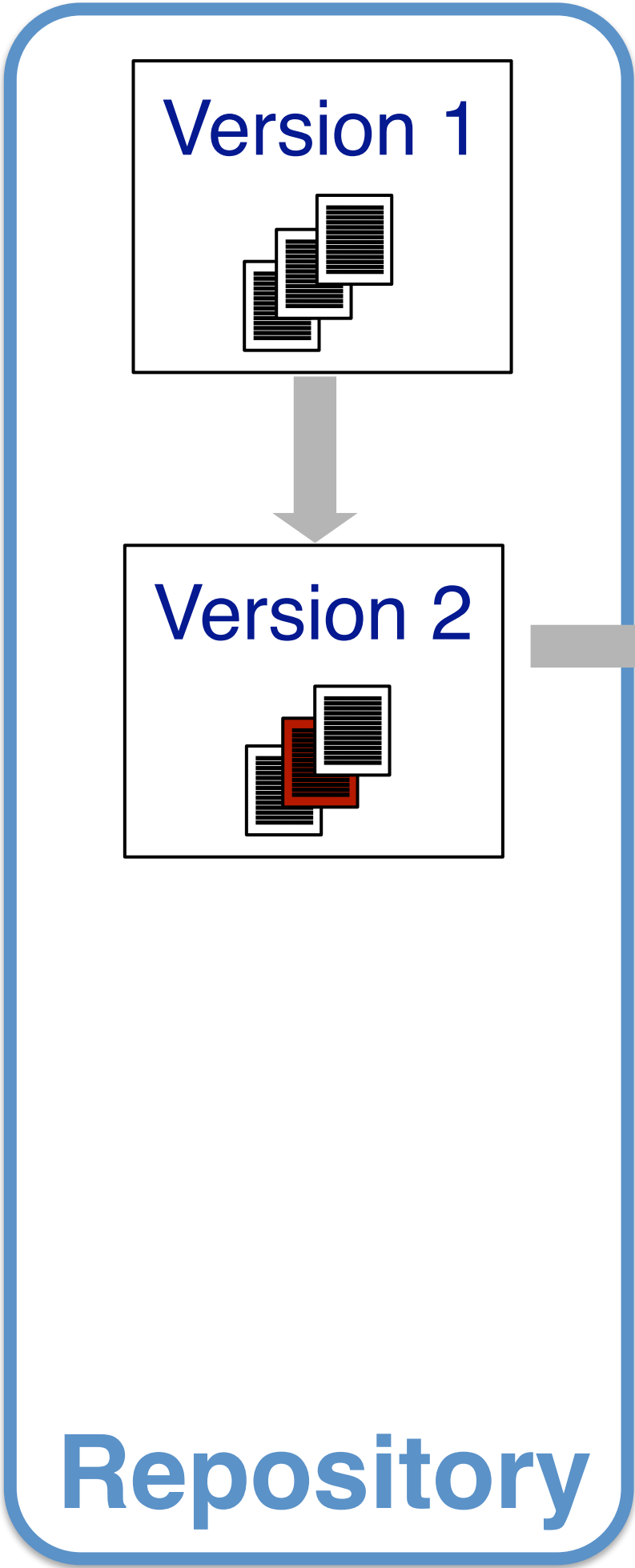
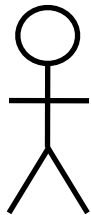


Repository

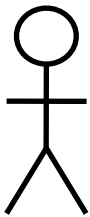
Carol



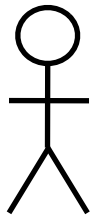
Bob



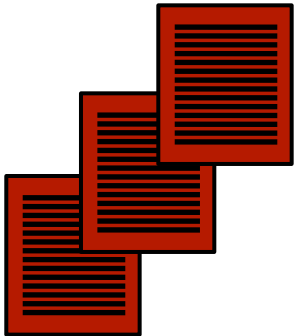
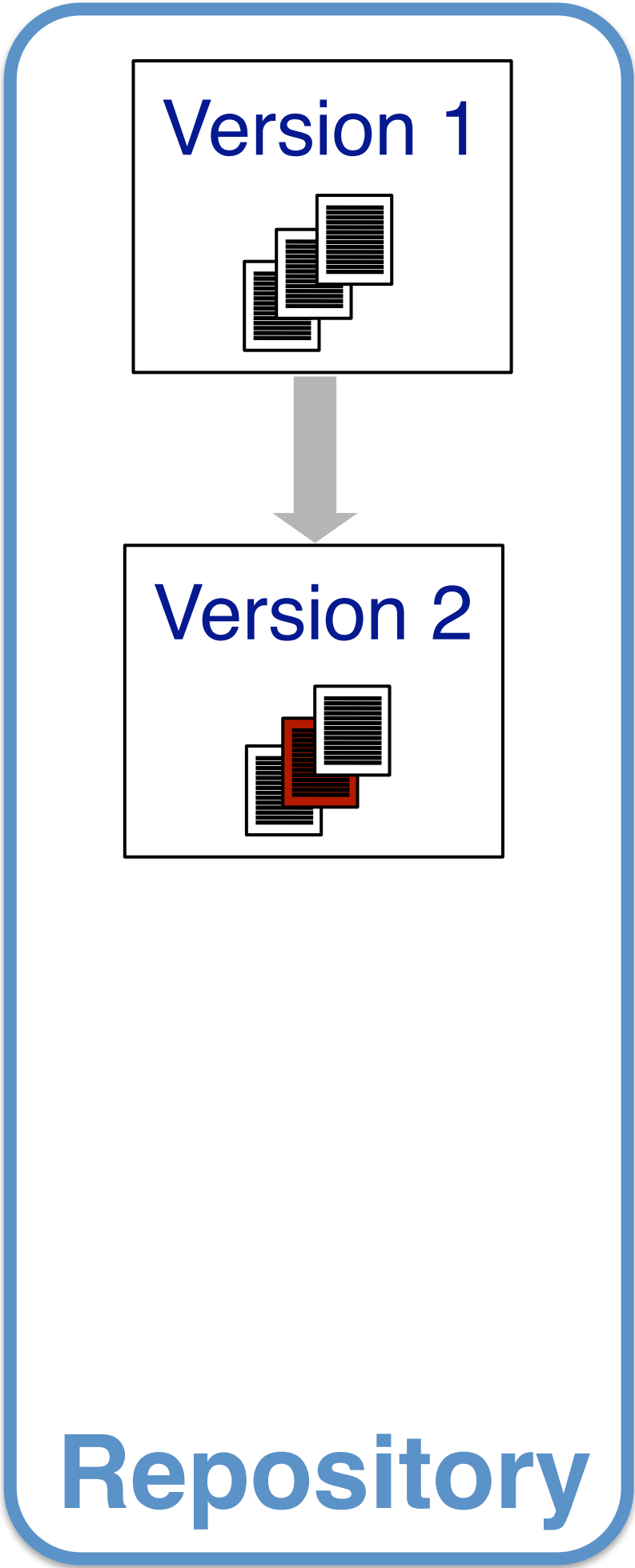
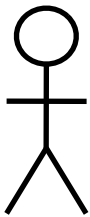
Carol



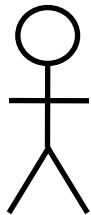
Bob



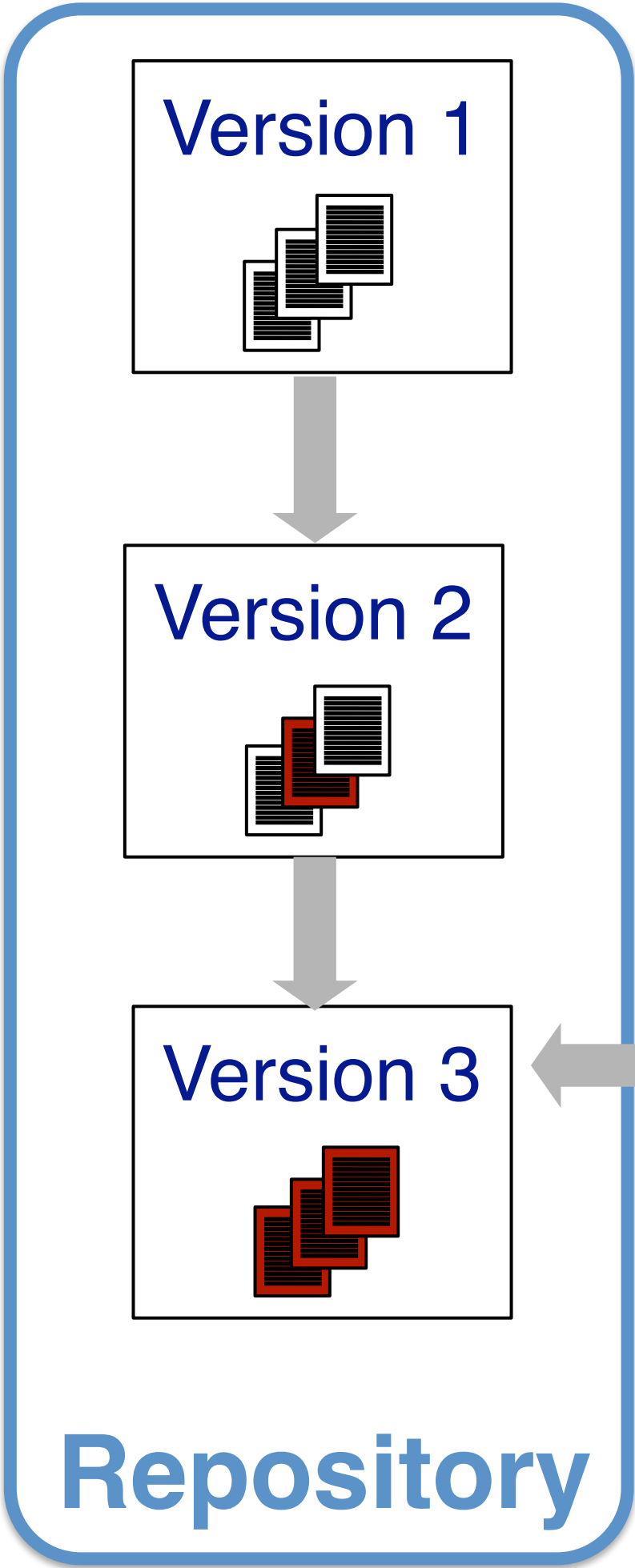
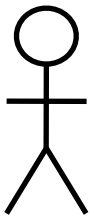
Carol



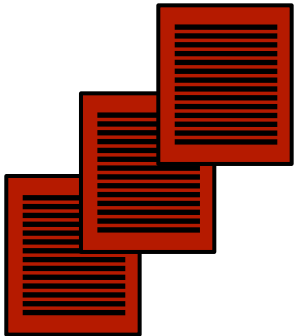
Bob



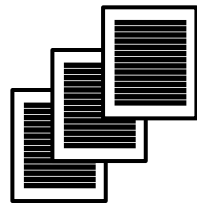
Carol



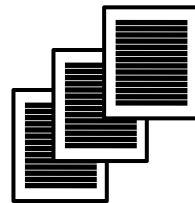
← Snapshot



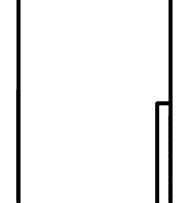
Version 1



Version 4

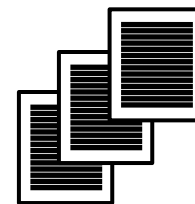


Version 3

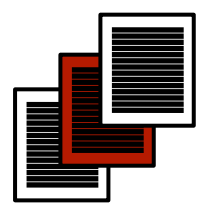


Version 99

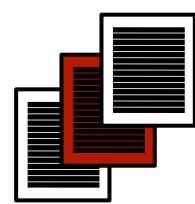
Version 100



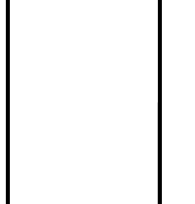
Version 2



Version 5

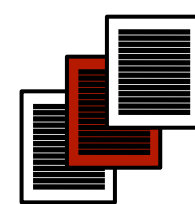


Version 3

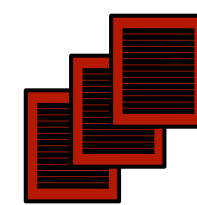


Version 99

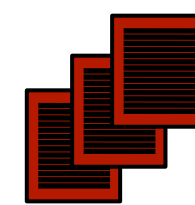
Version 101



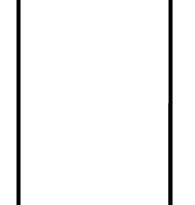
Version 3



Version 6

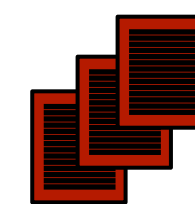


Version 3



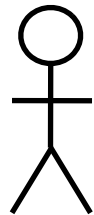
Version 99

Version 103

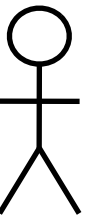


Repository

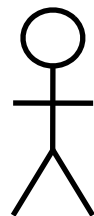
Bob



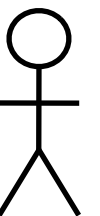
Carol



Alice



Ted



Version 1

Version 4

Version 3

Version 6

Version 100

Version 2

Version 5

Version 7

Version 8

Version 101

Version 103

Controlled evolution

Can still lead to disaster!

Repository

Bob



Carol



Ted



Alice



git

git

Tracks the history of a collection of files

git

Tracks the history of a collection of files

Can revert the collection of files to another version

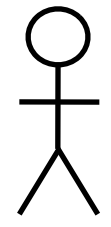
git

**distributed
version control
system**

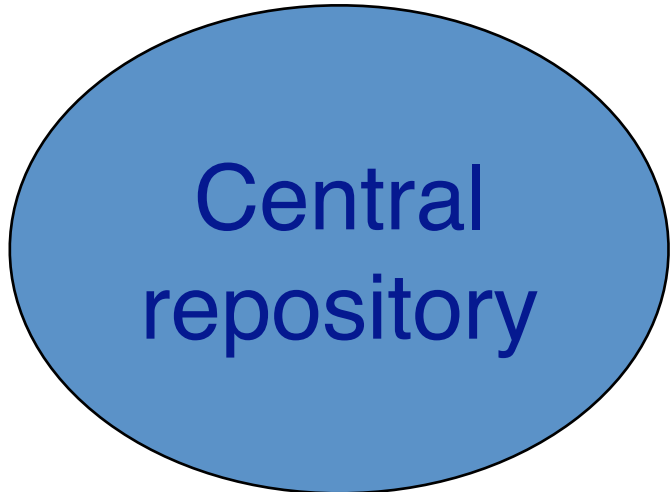
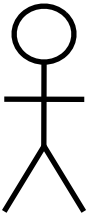
**What is a
distributed version
control system?**

**What is a
centralized version
control system?**

Bob

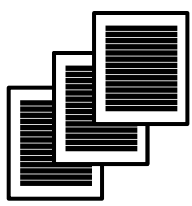
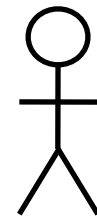


Carol

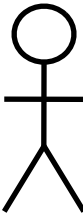


Central
repository

Bob

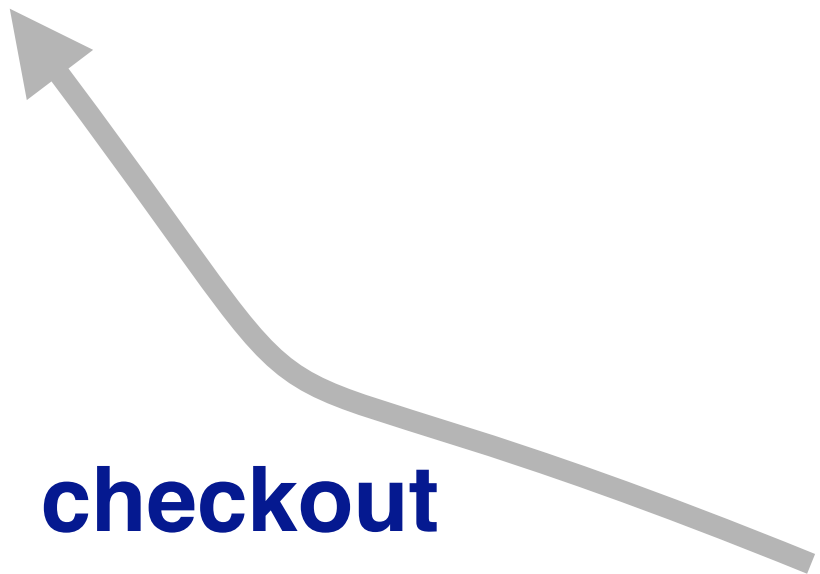
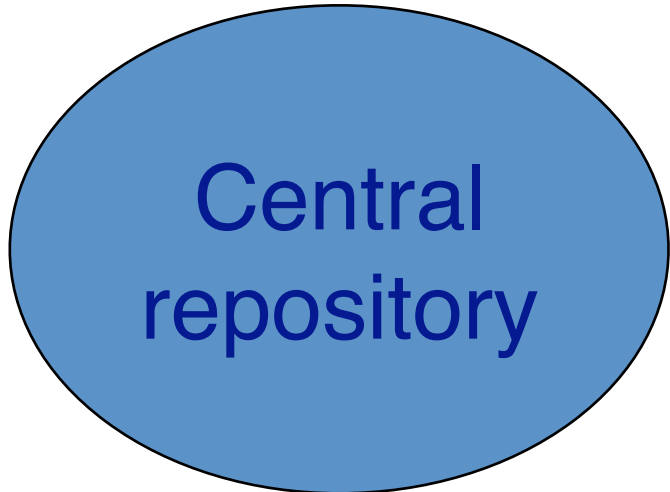


Carol

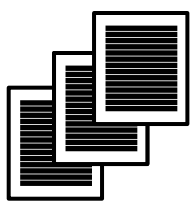
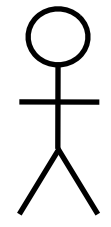


checkout

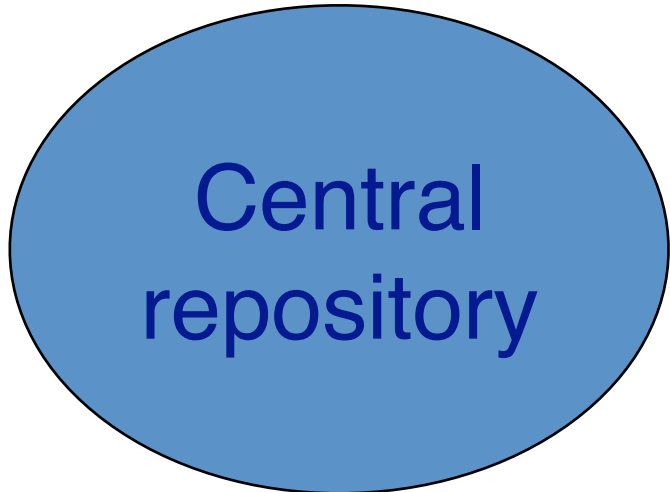
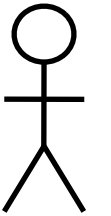
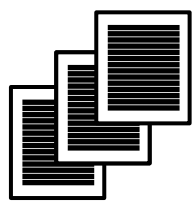
Central
repository



Bob

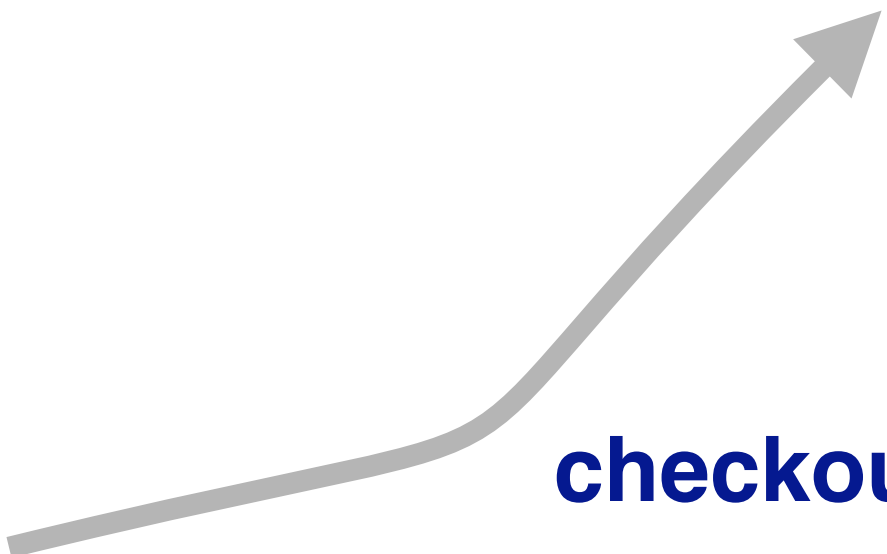
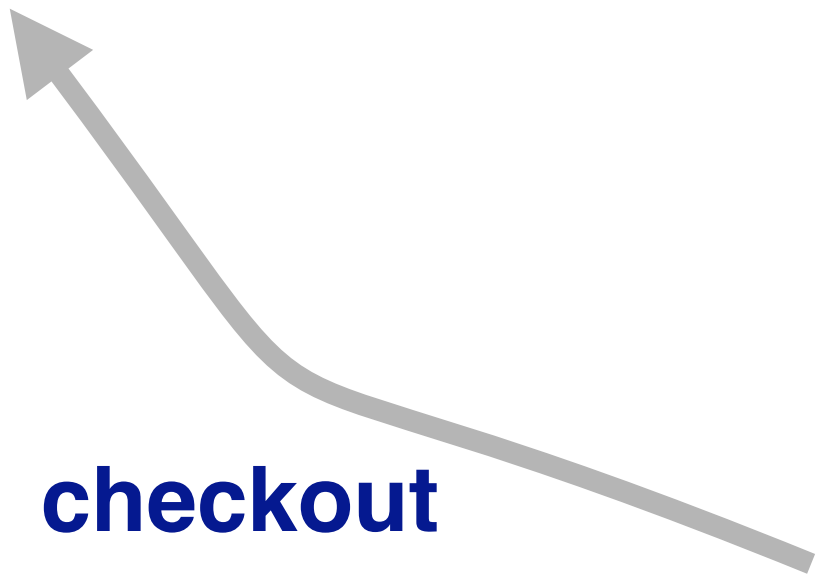


Carol

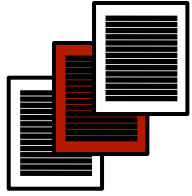
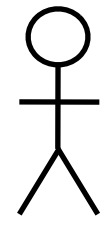


checkout

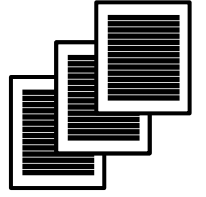
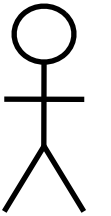
checkout



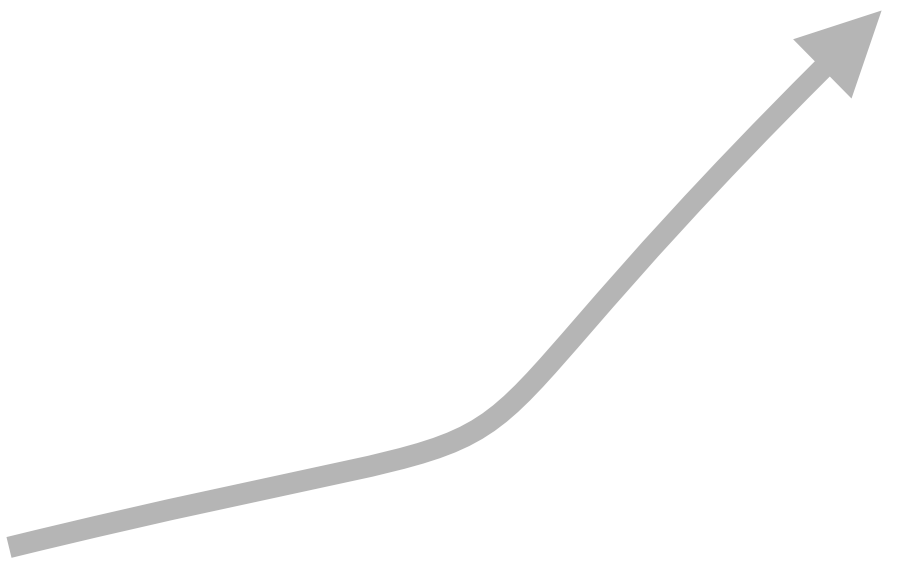
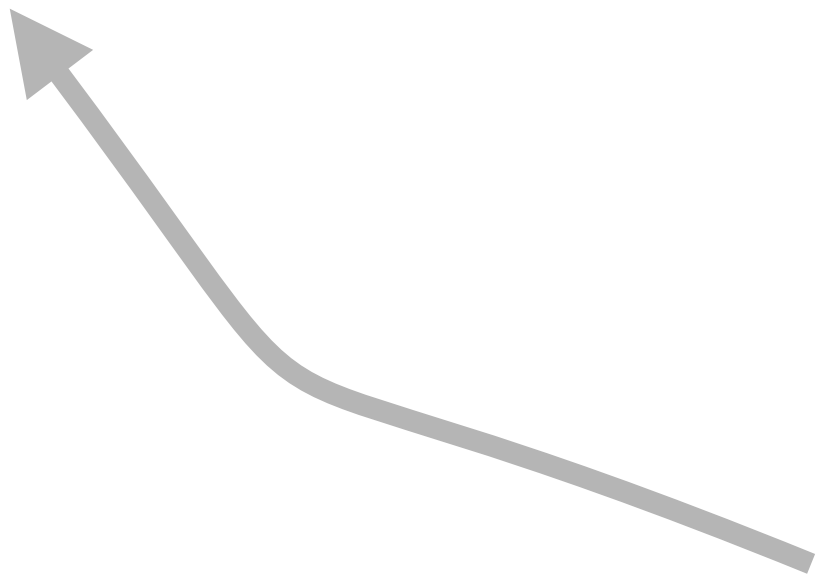
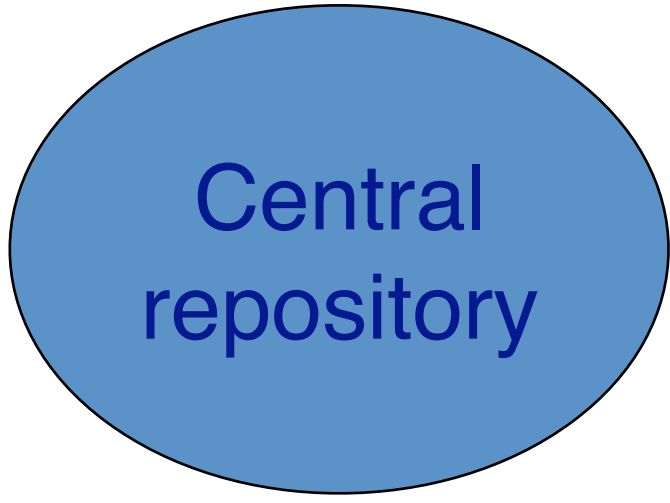
Bob

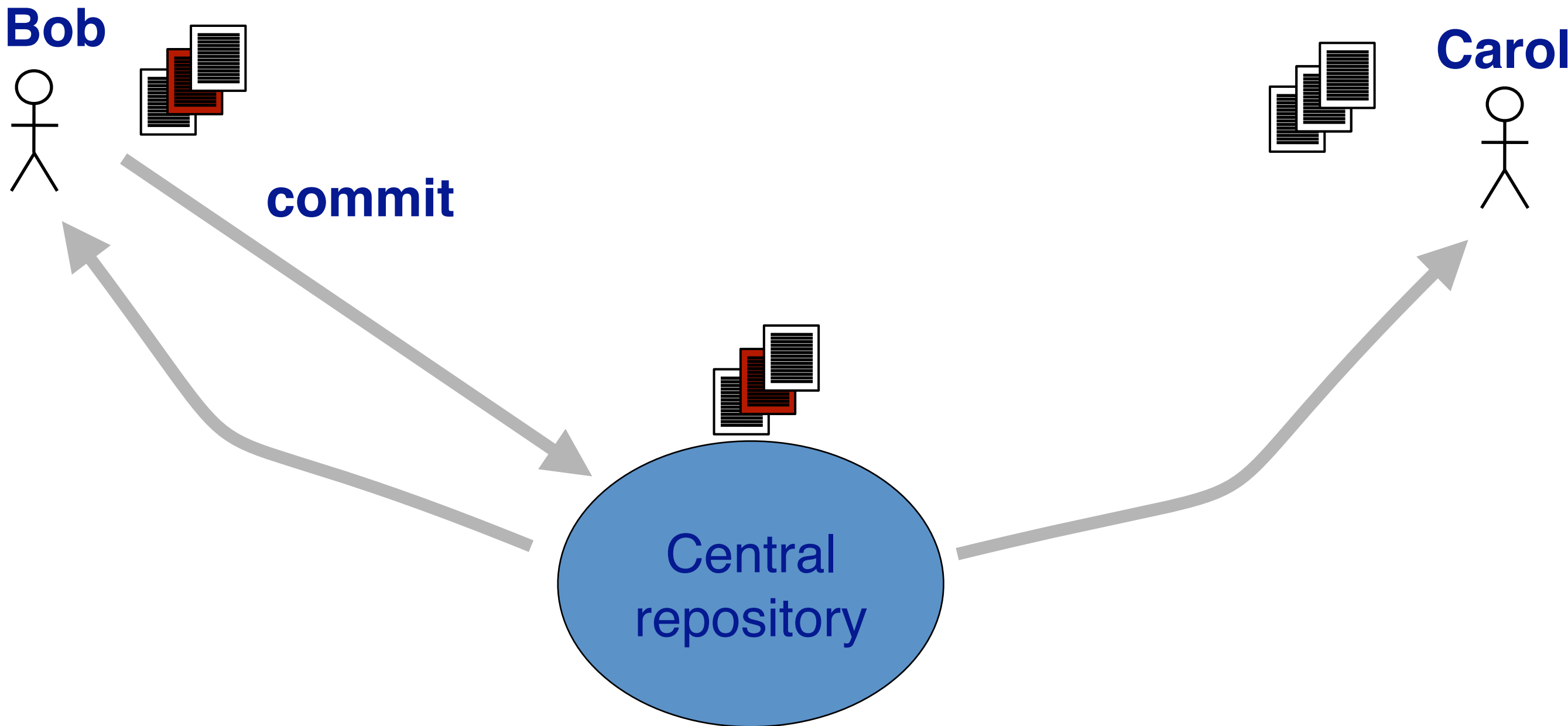


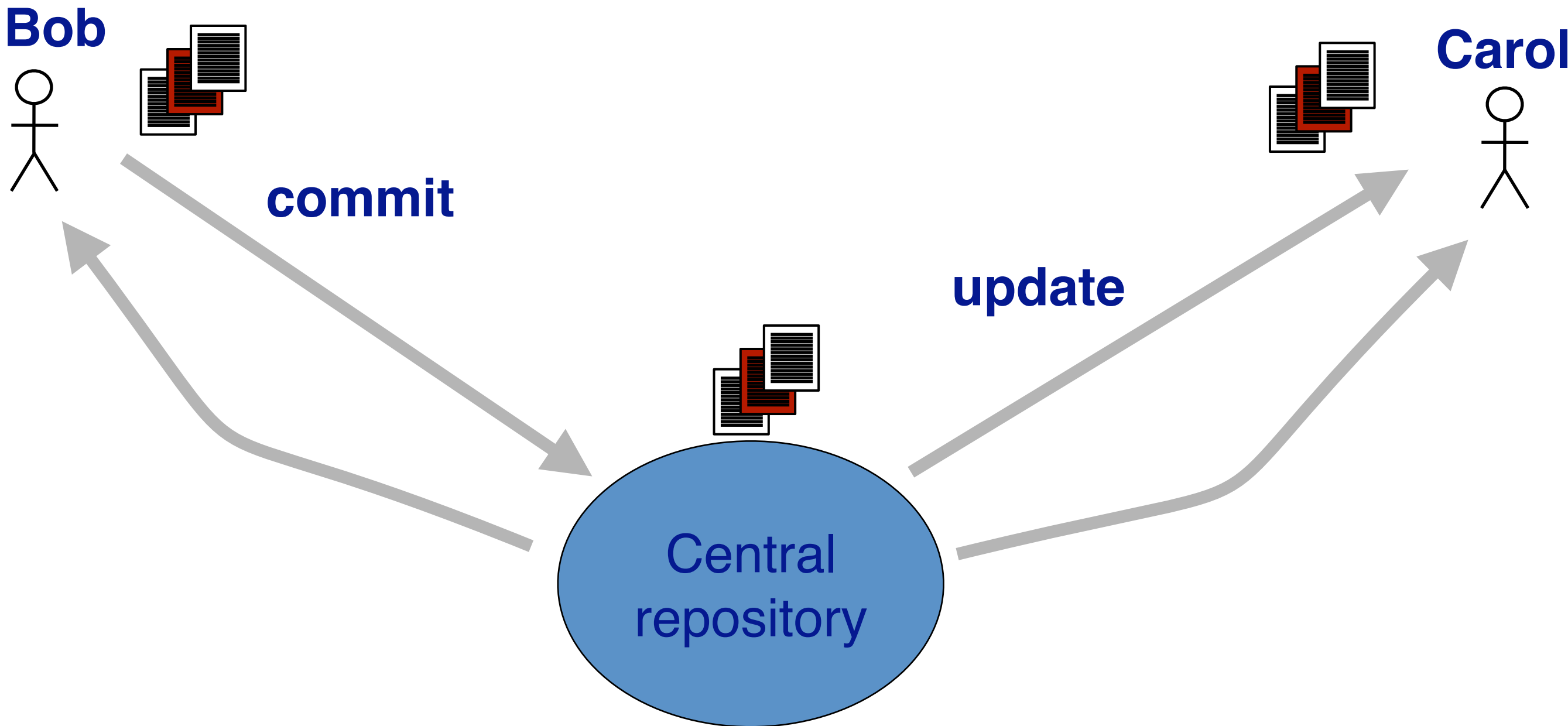
Carol

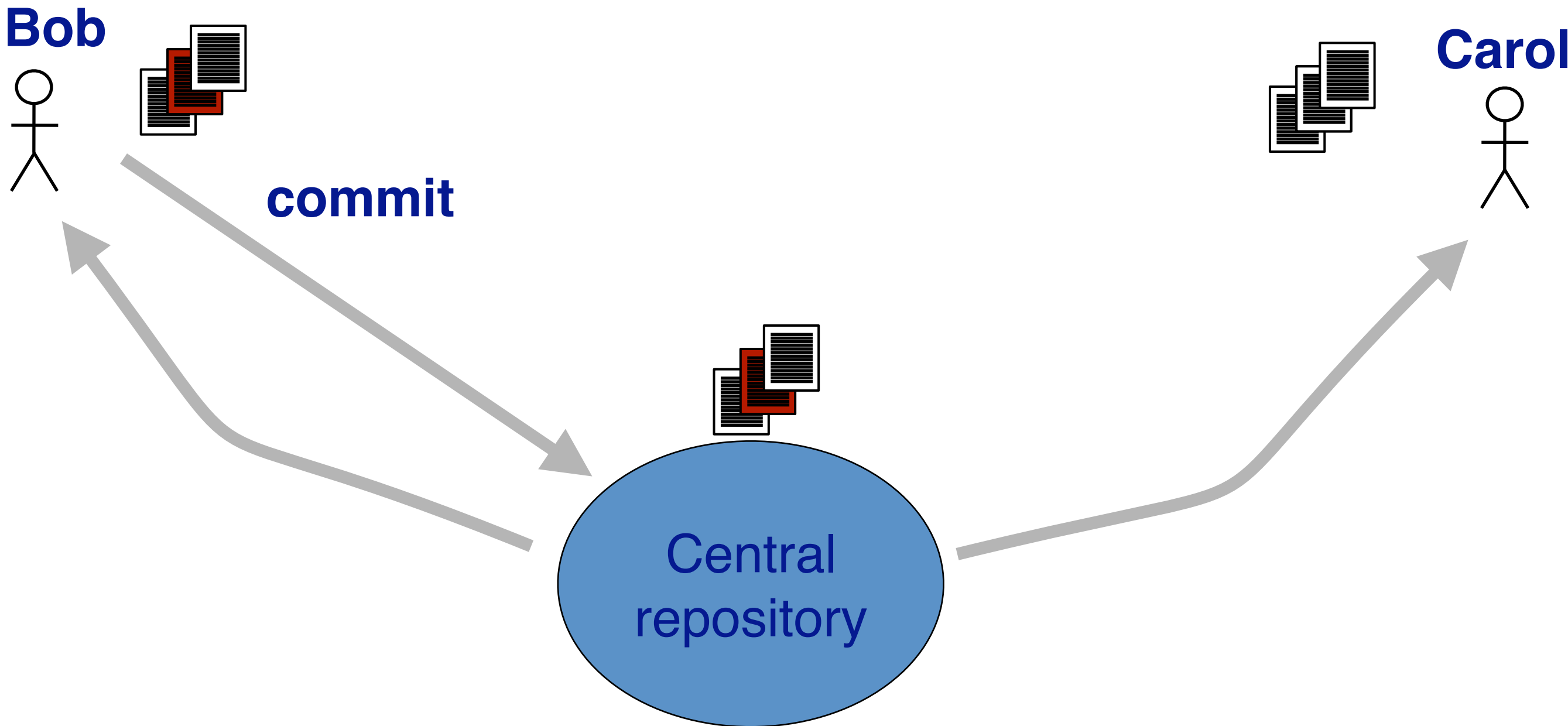


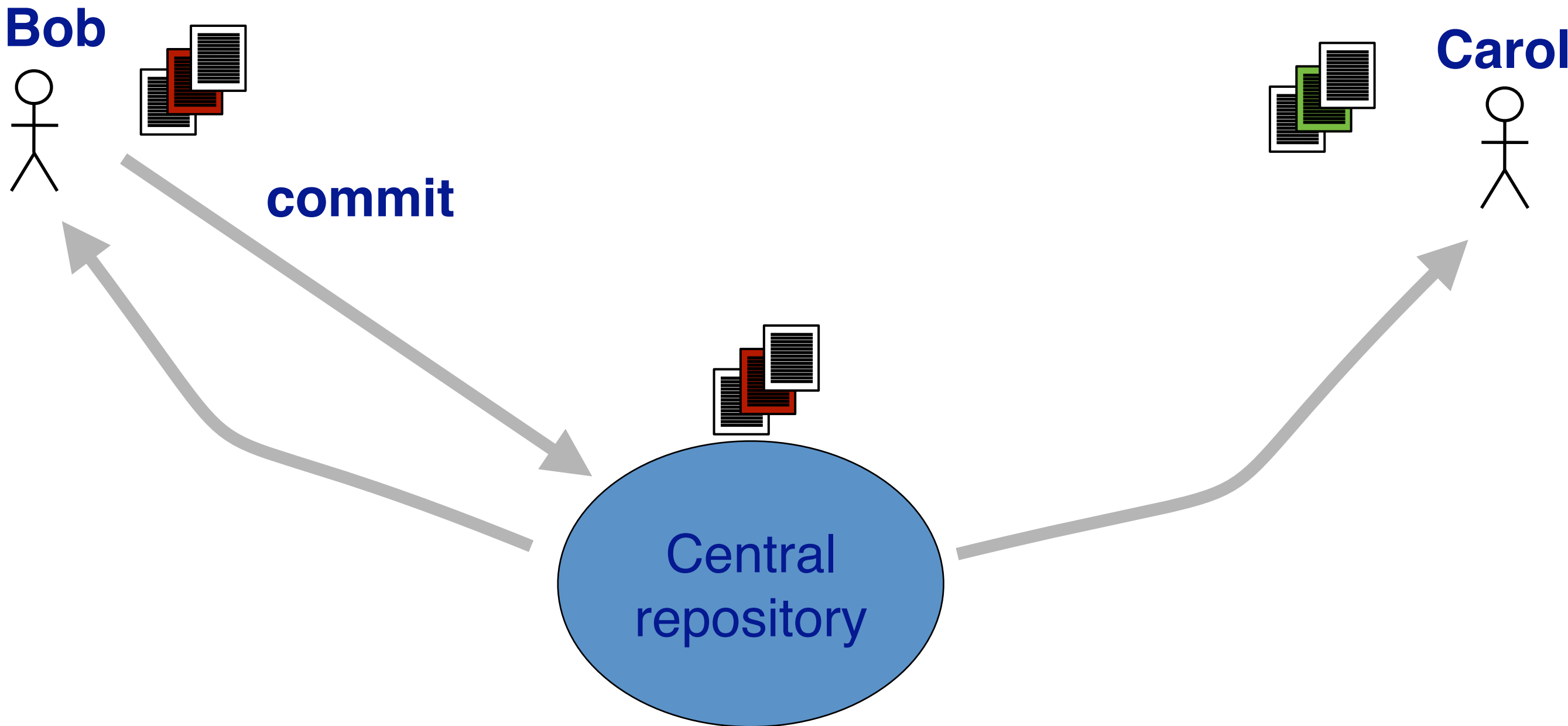
Central
repository

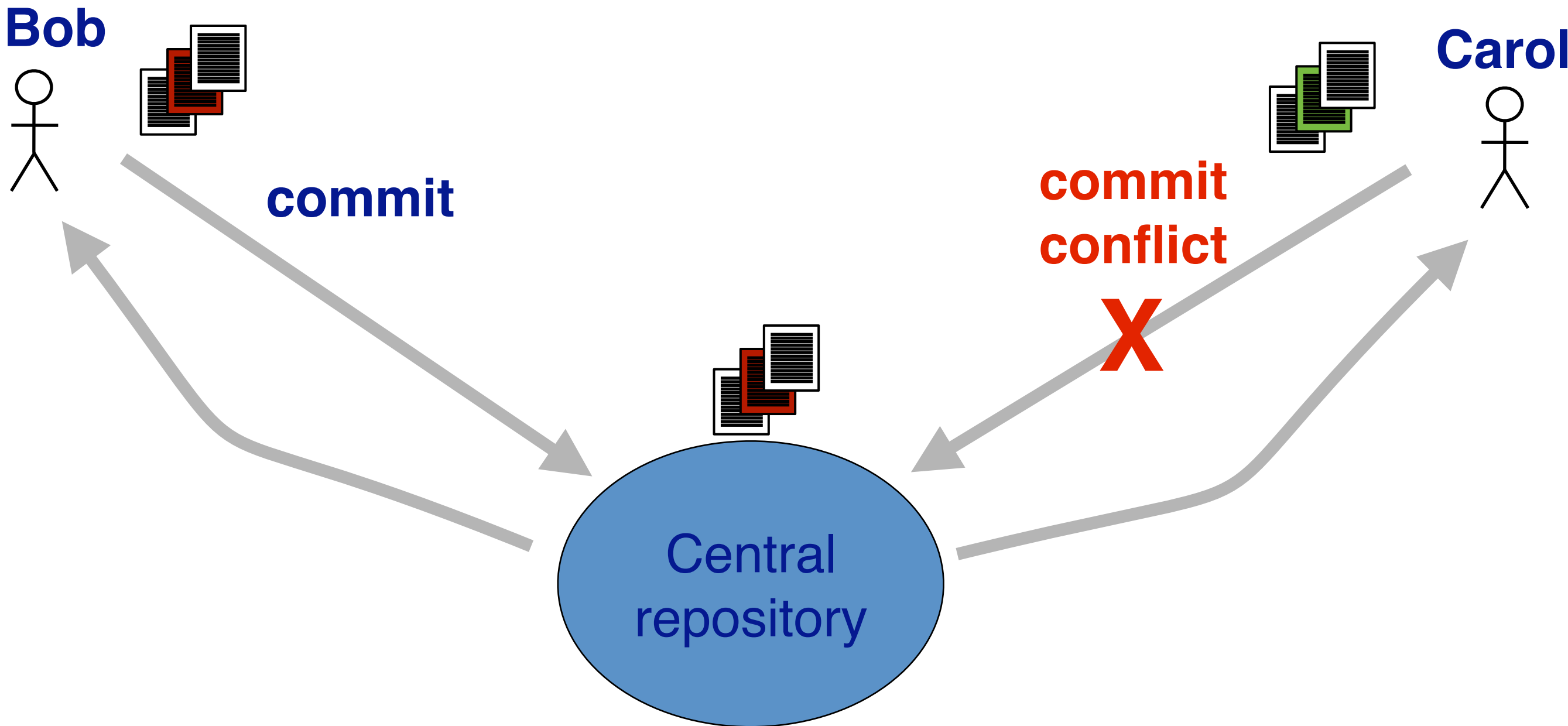


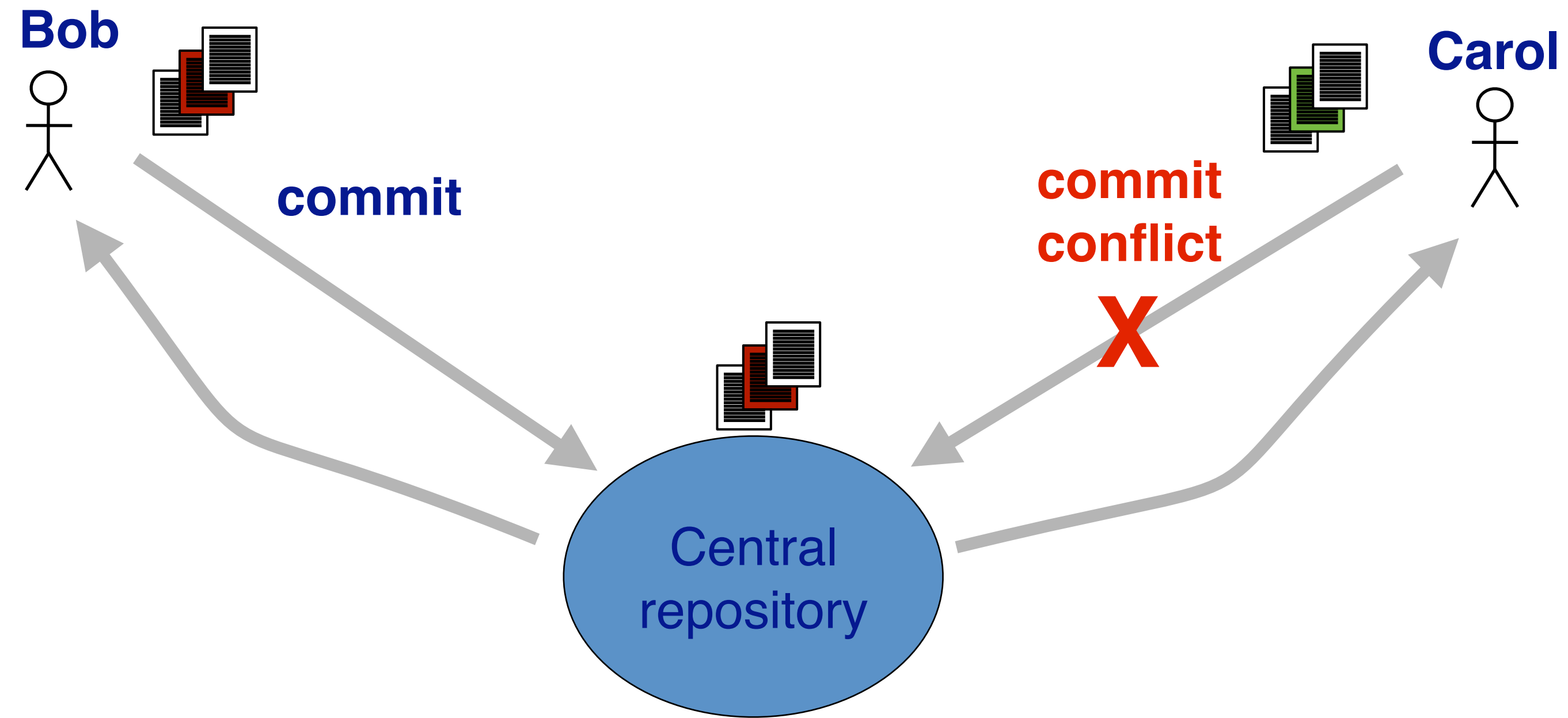








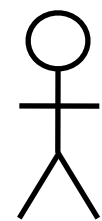




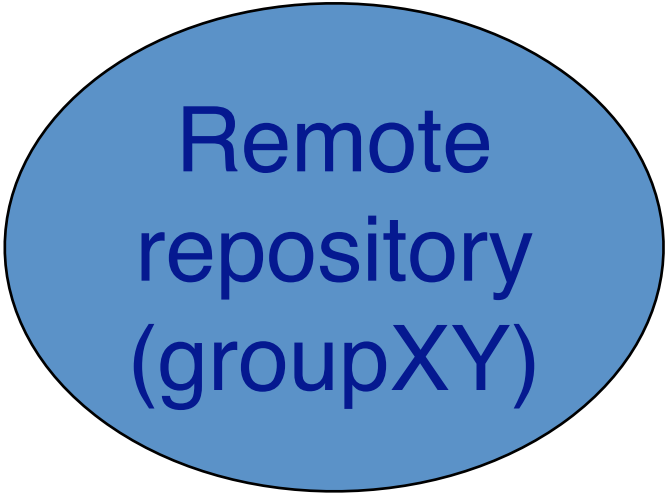
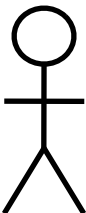
you must update before every commit

**What is a
distributed version
control system?**

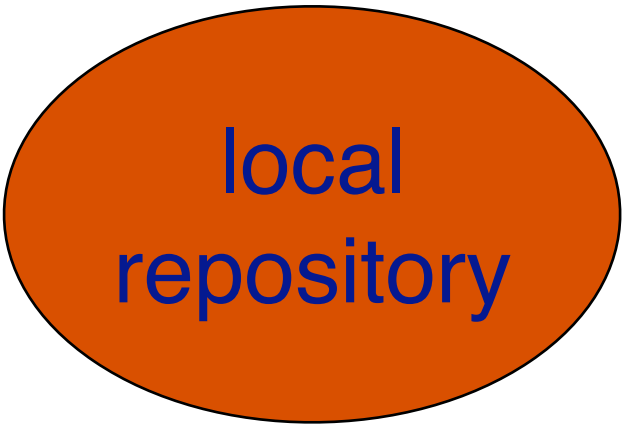
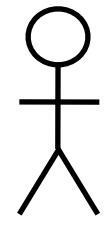
Bob



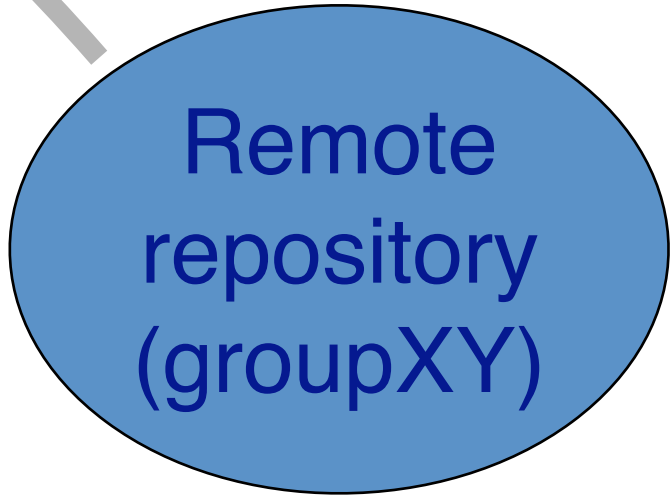
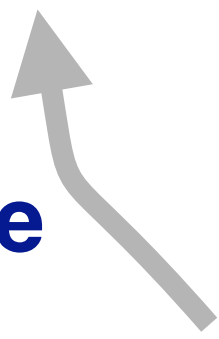
Carol



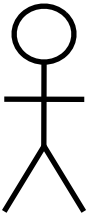
Bob



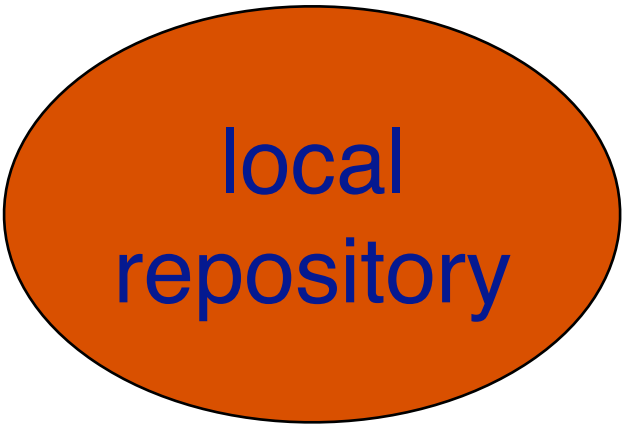
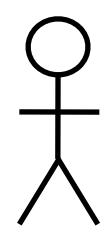
clone



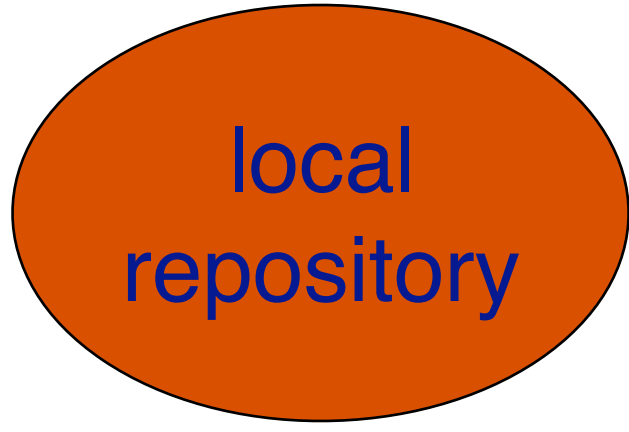
Carol



Bob

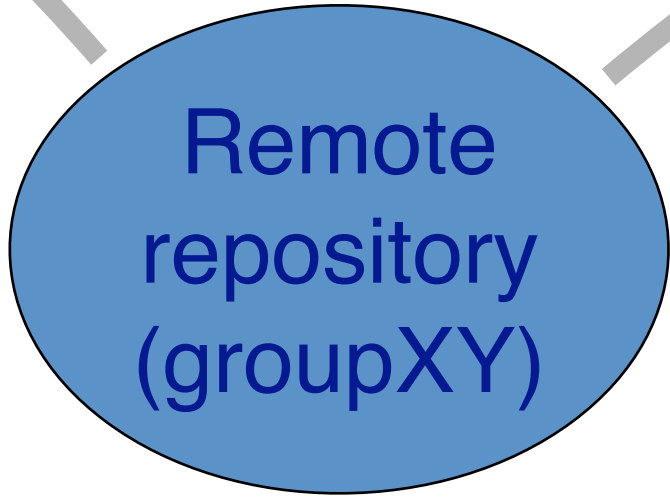
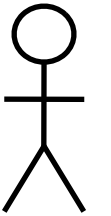


clone

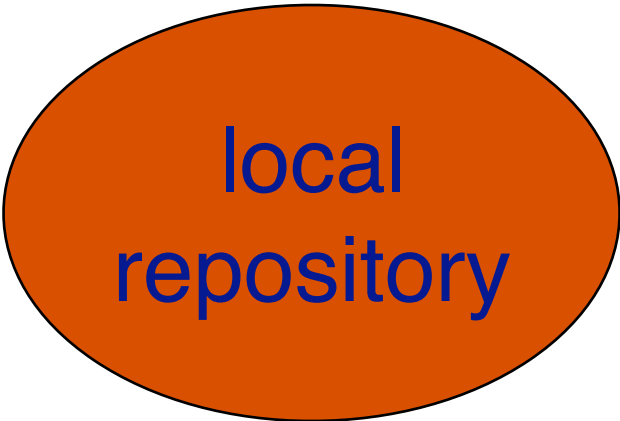
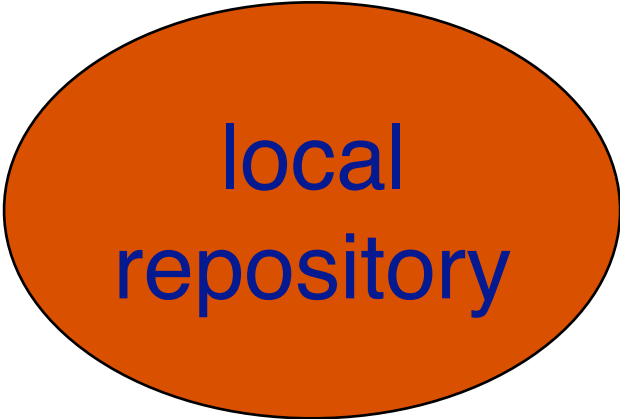
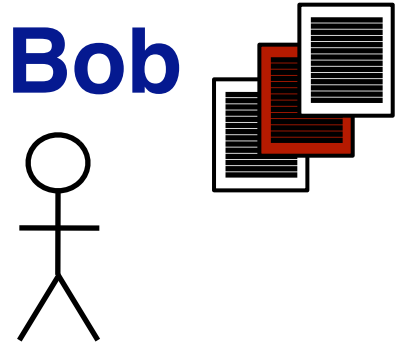


clone

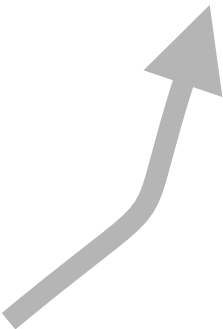
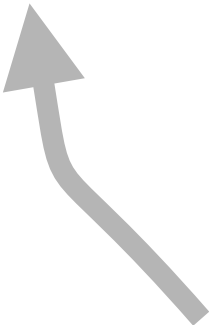
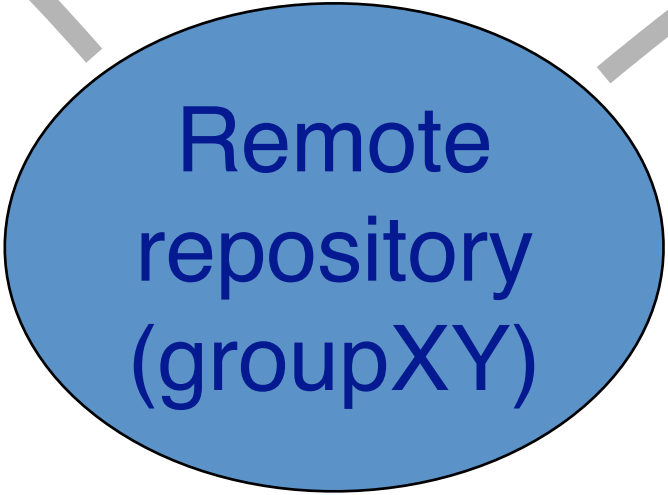
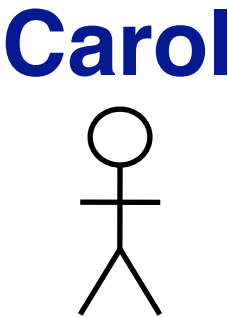
Carol

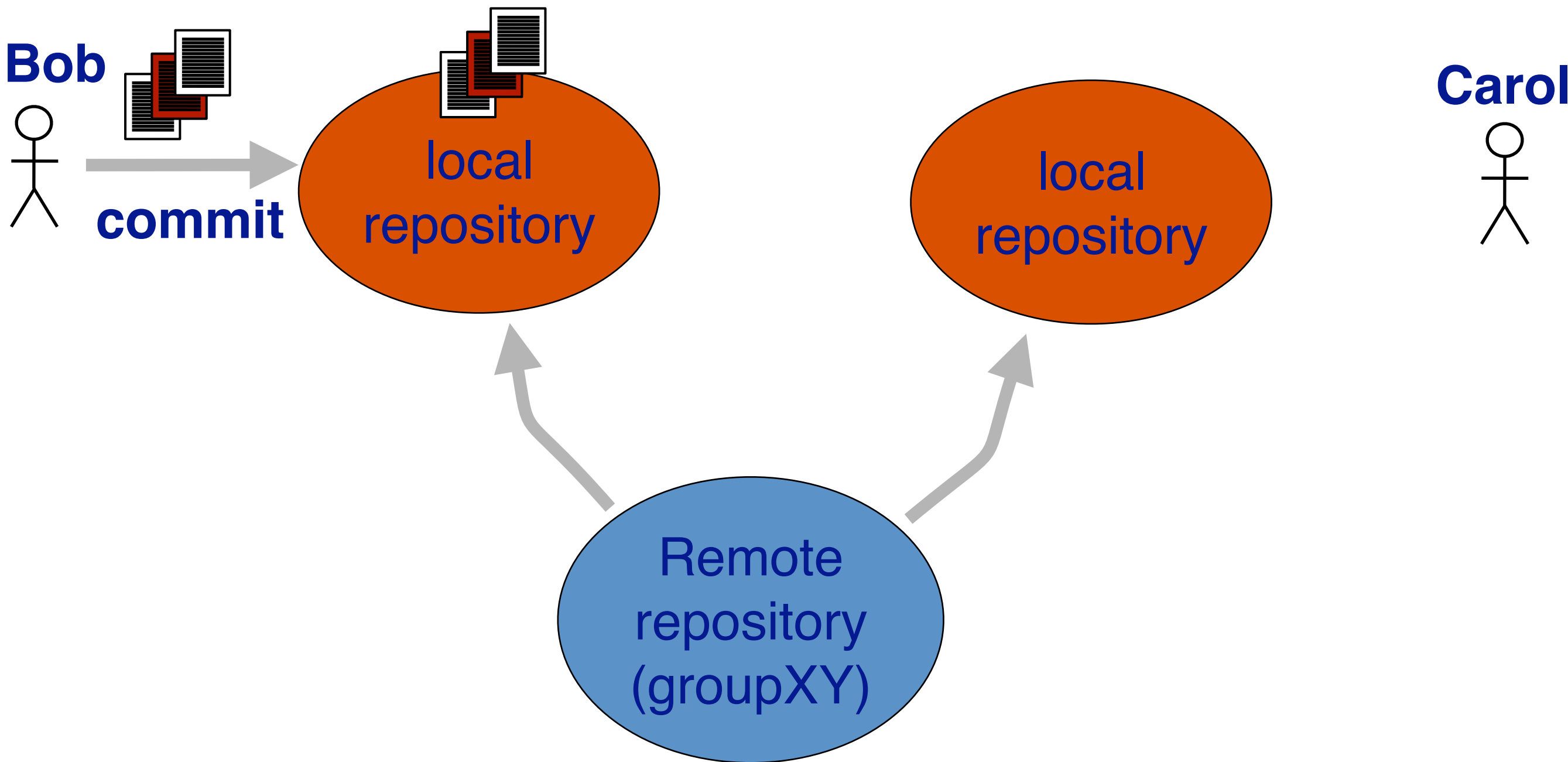


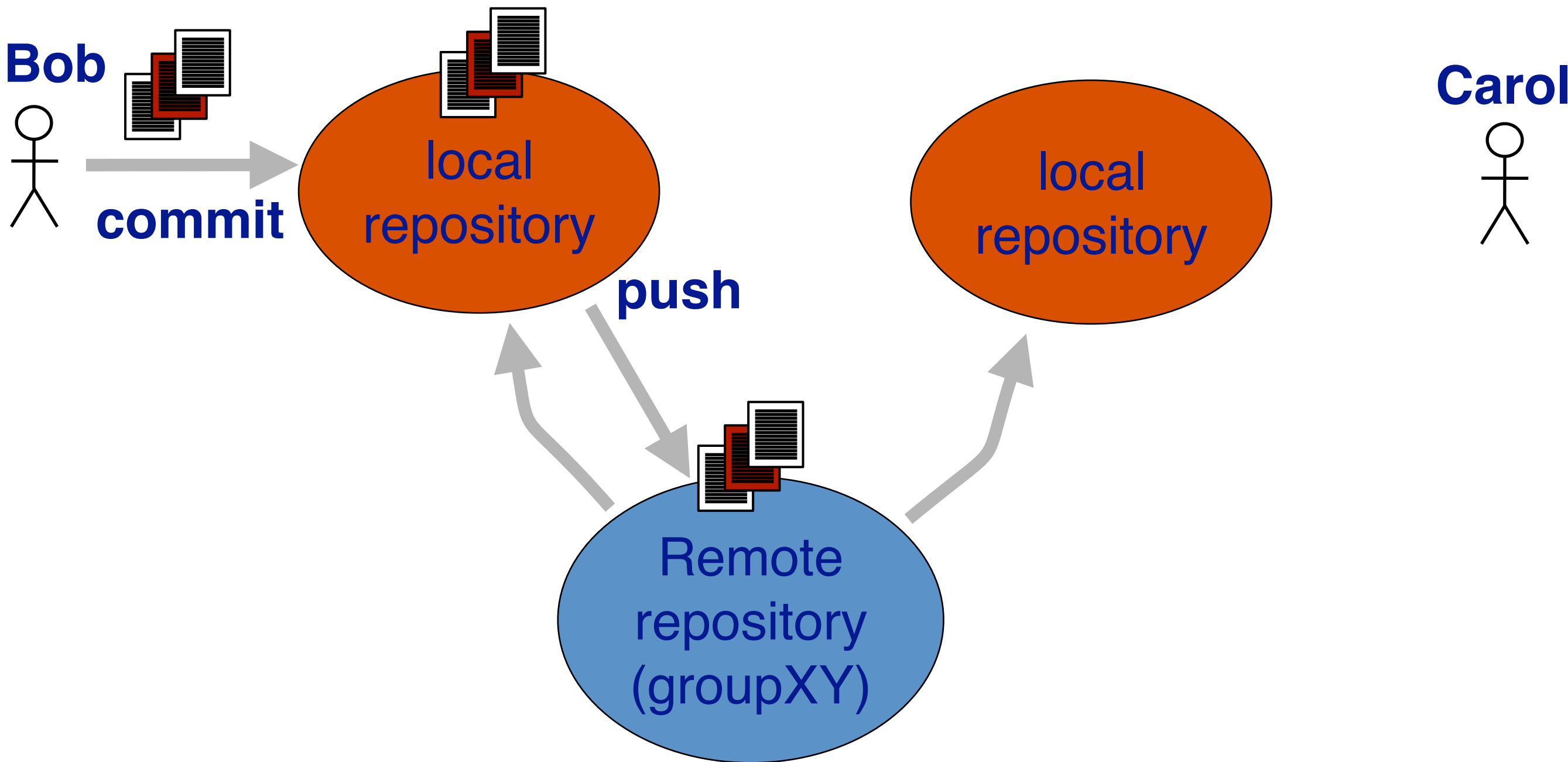
Bob

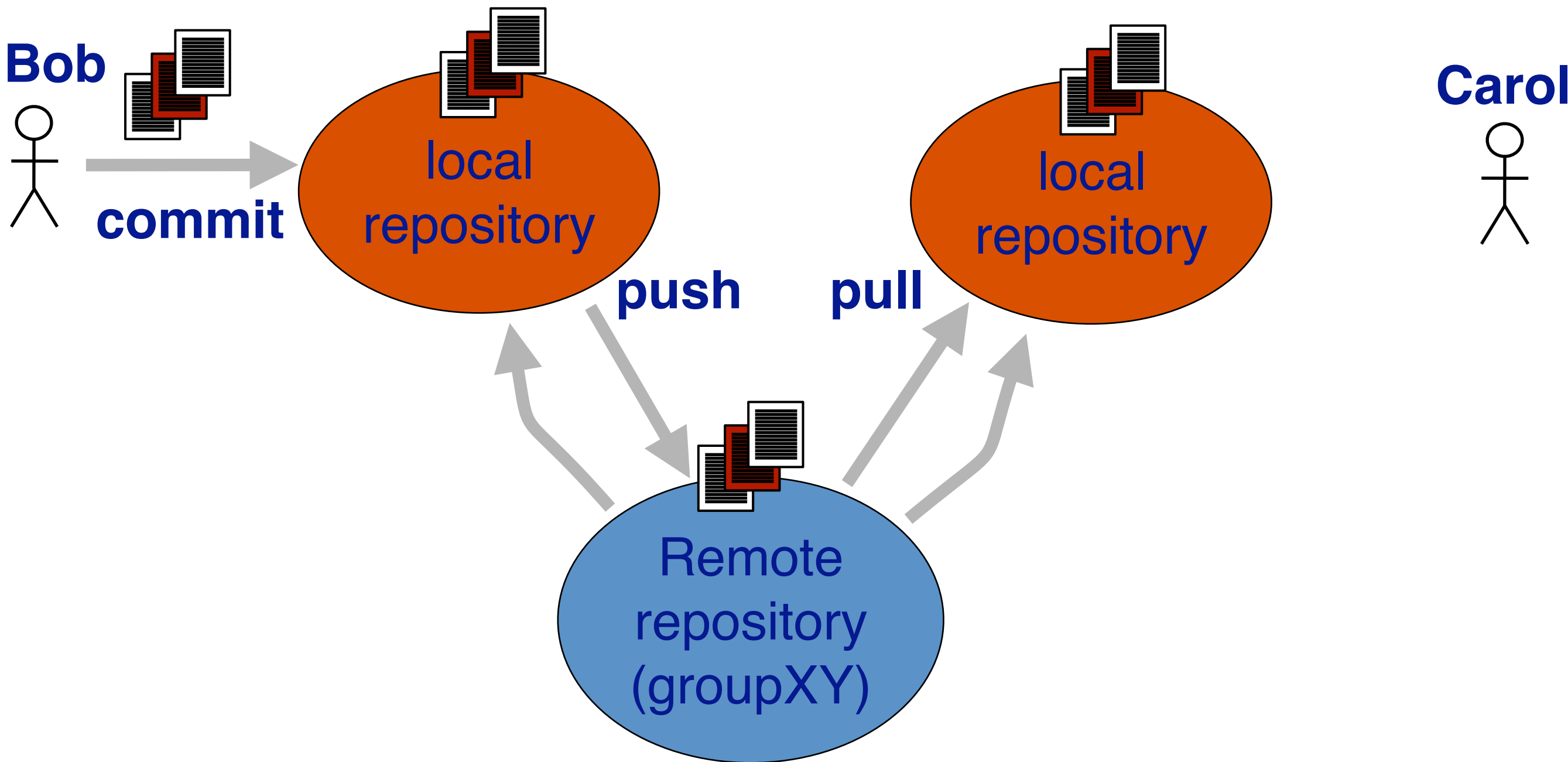


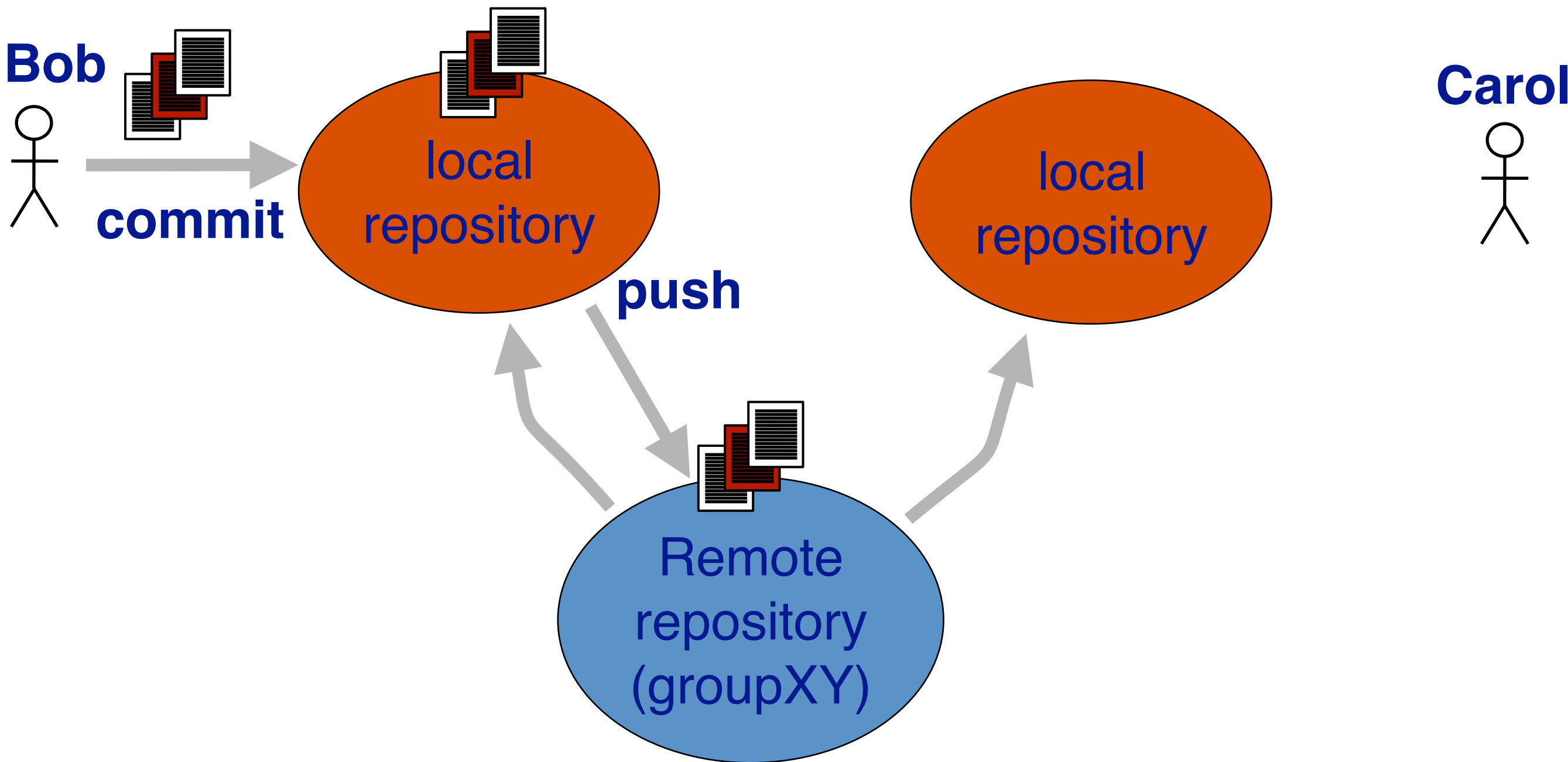
Carol

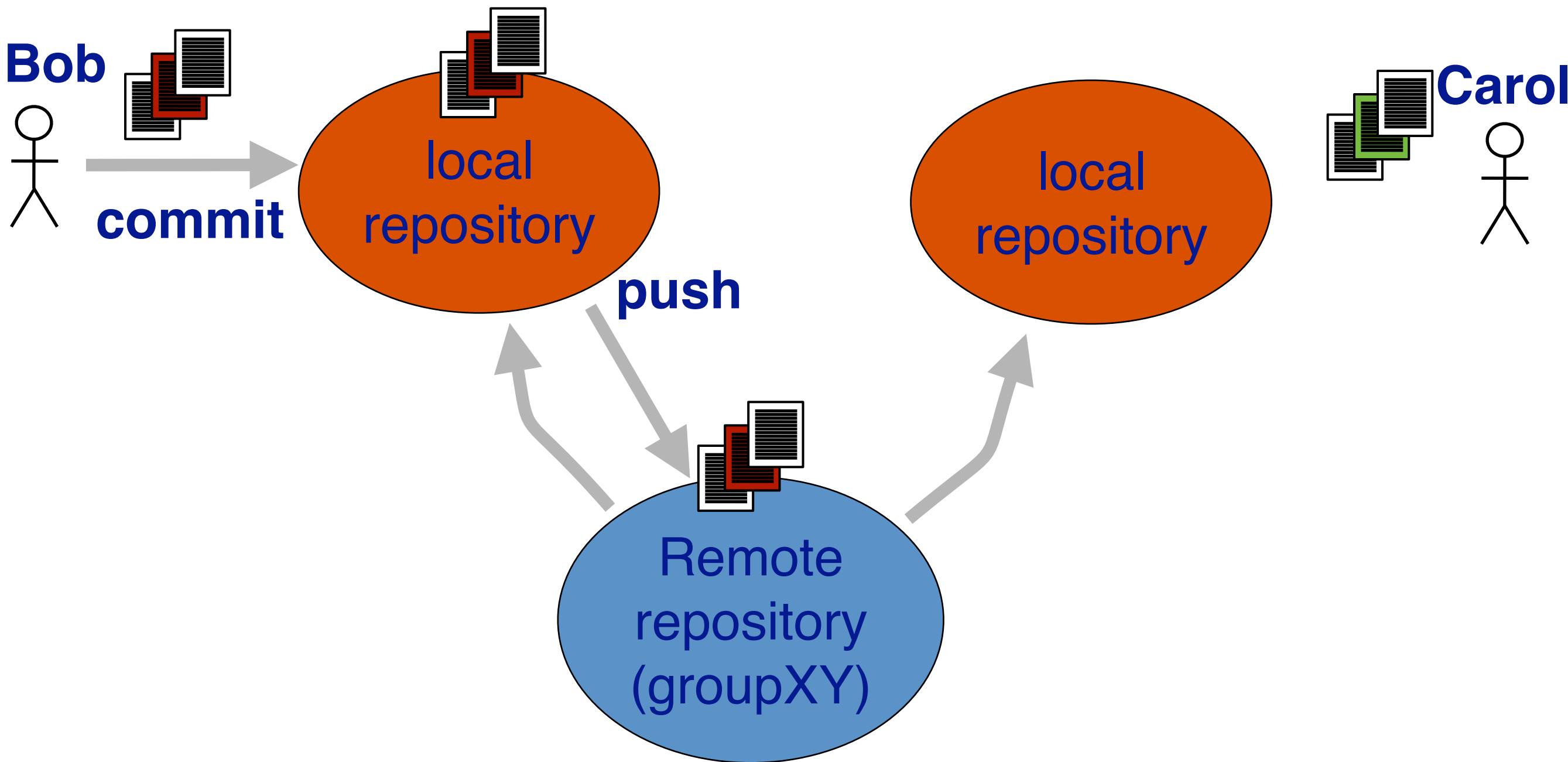


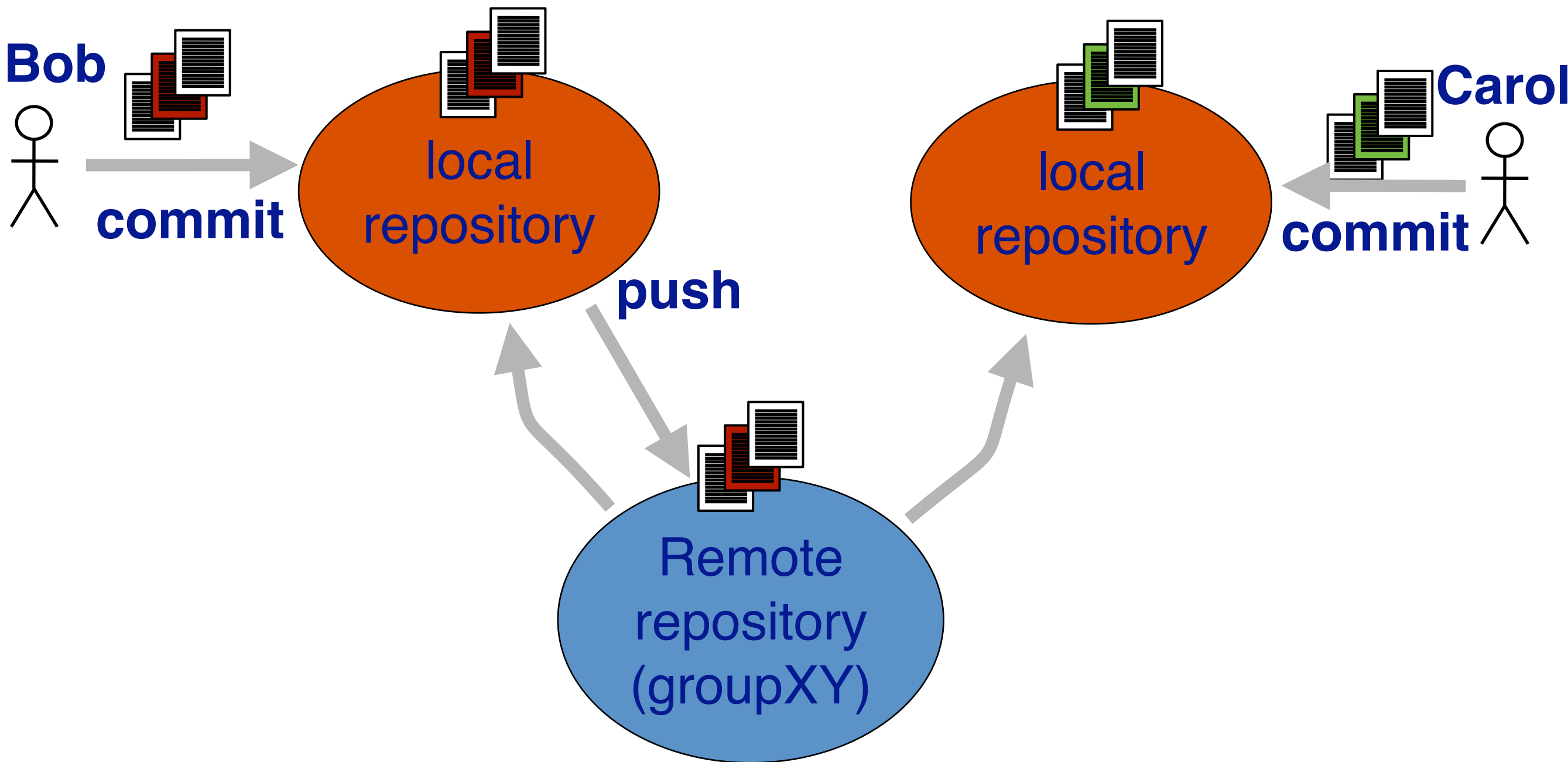


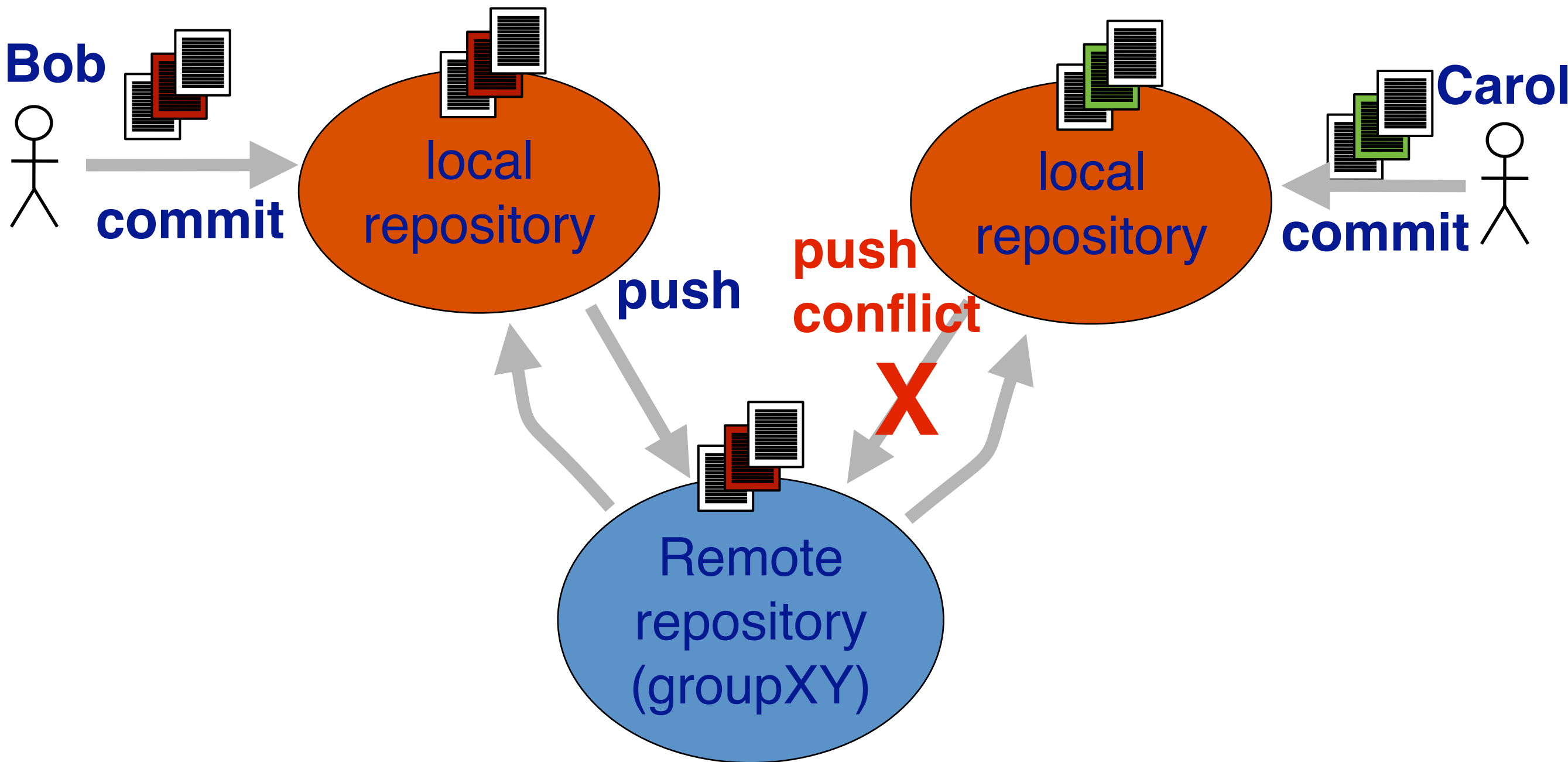


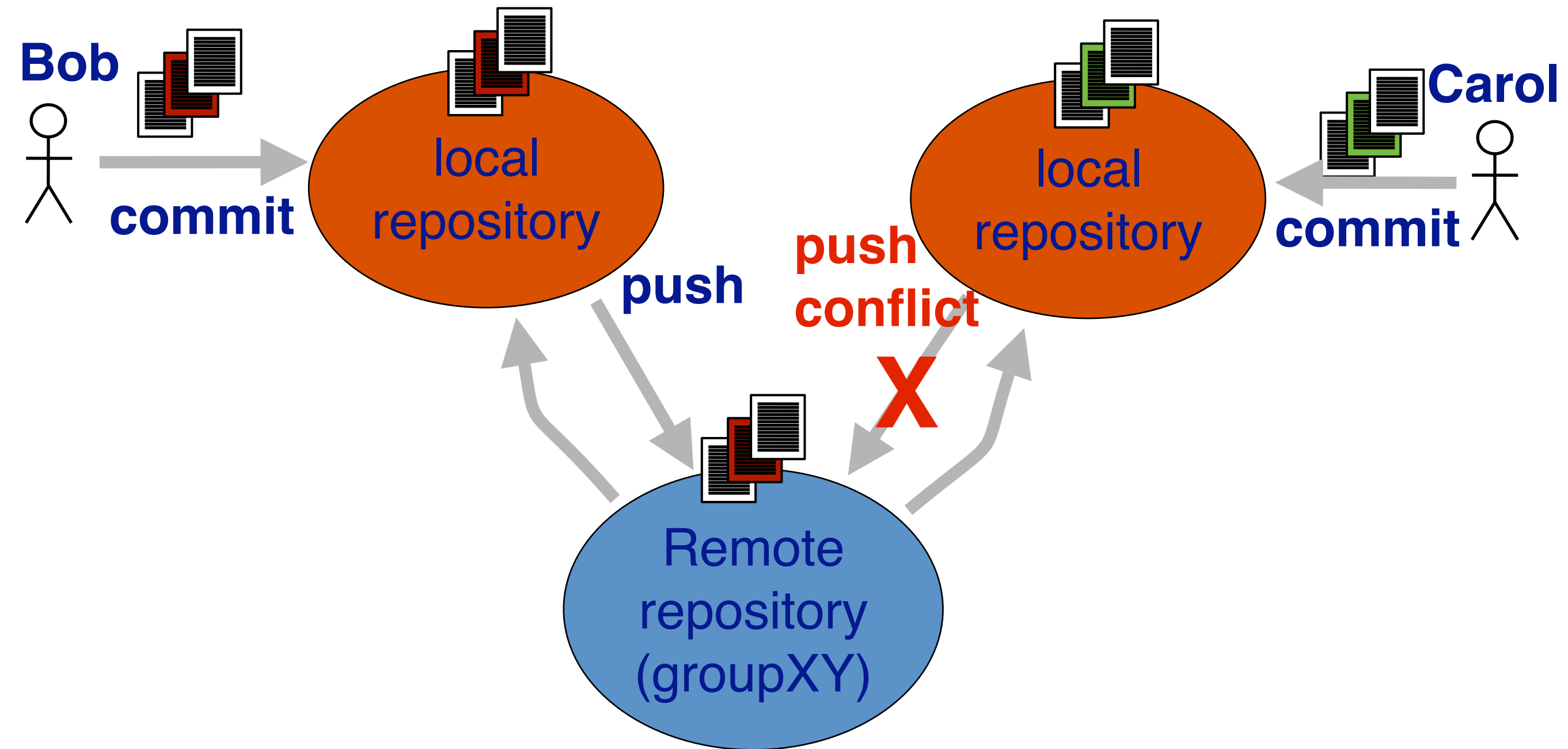




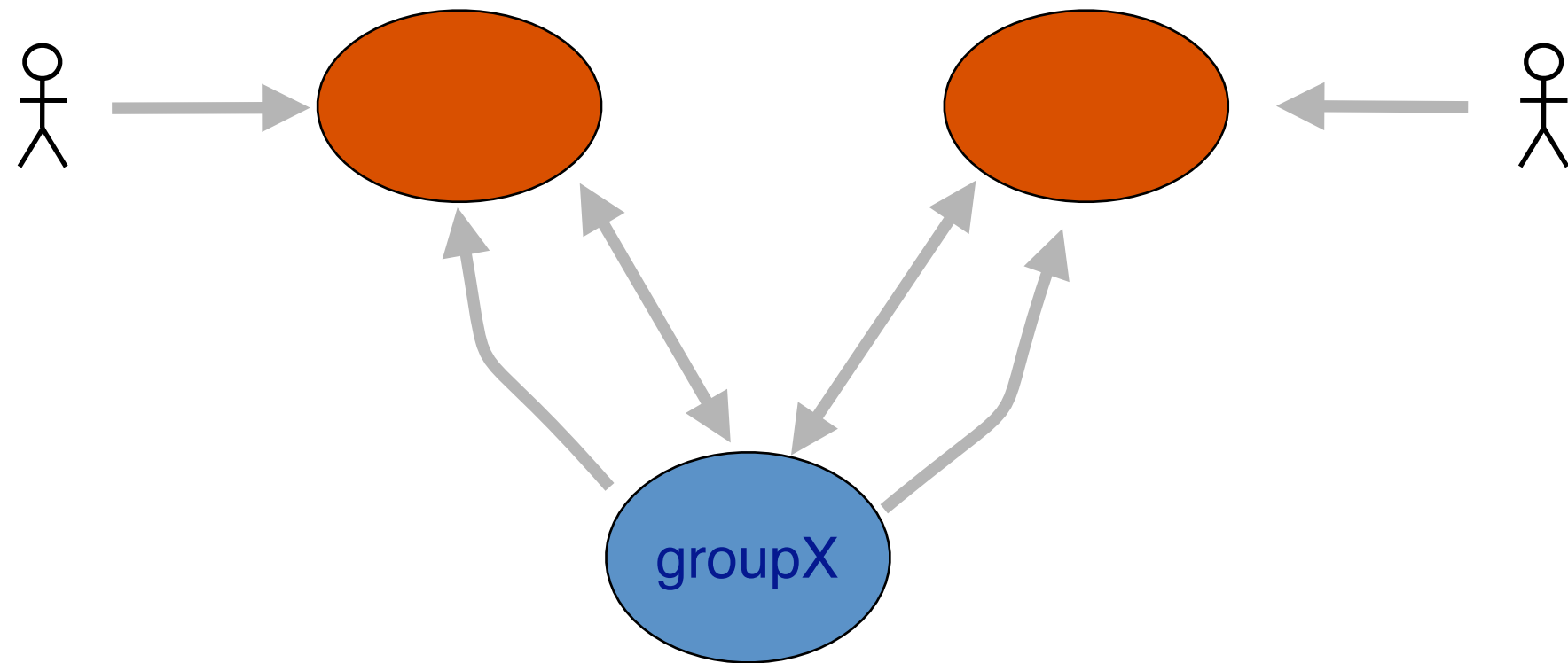


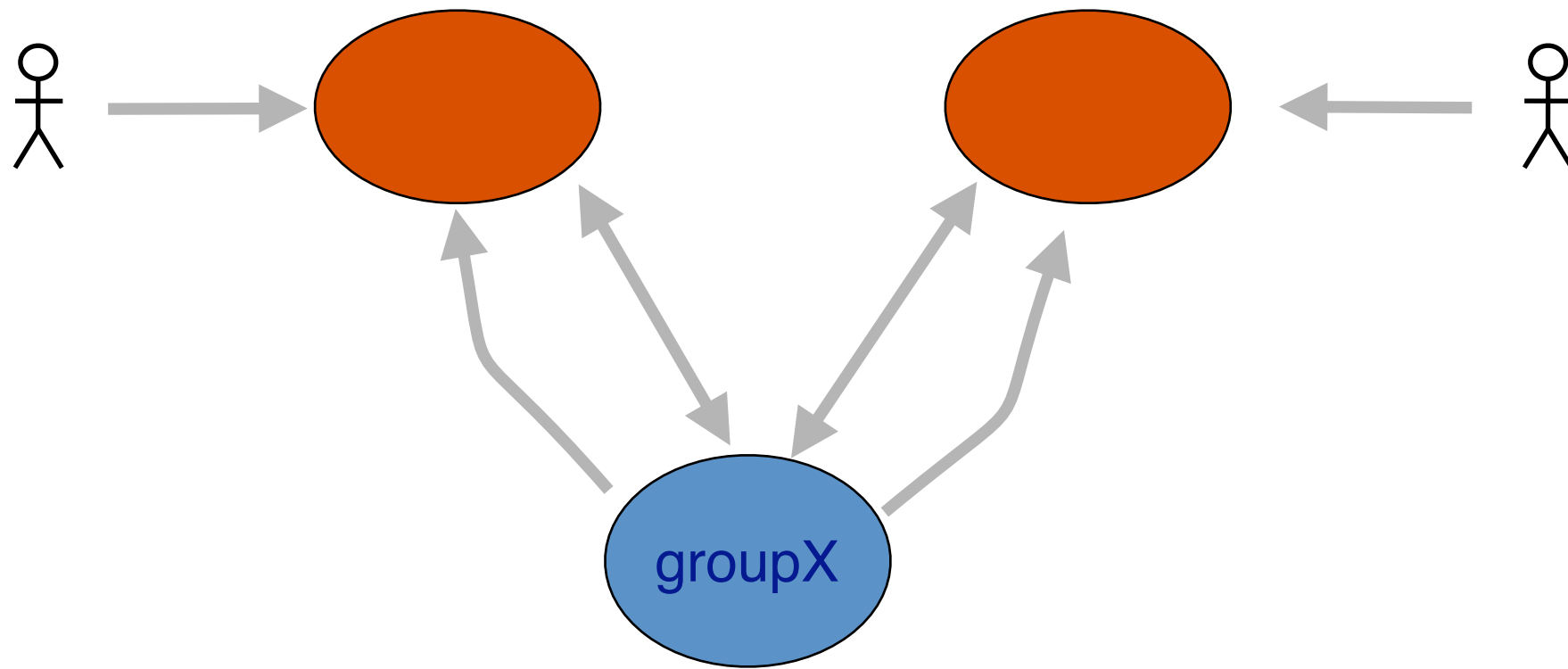




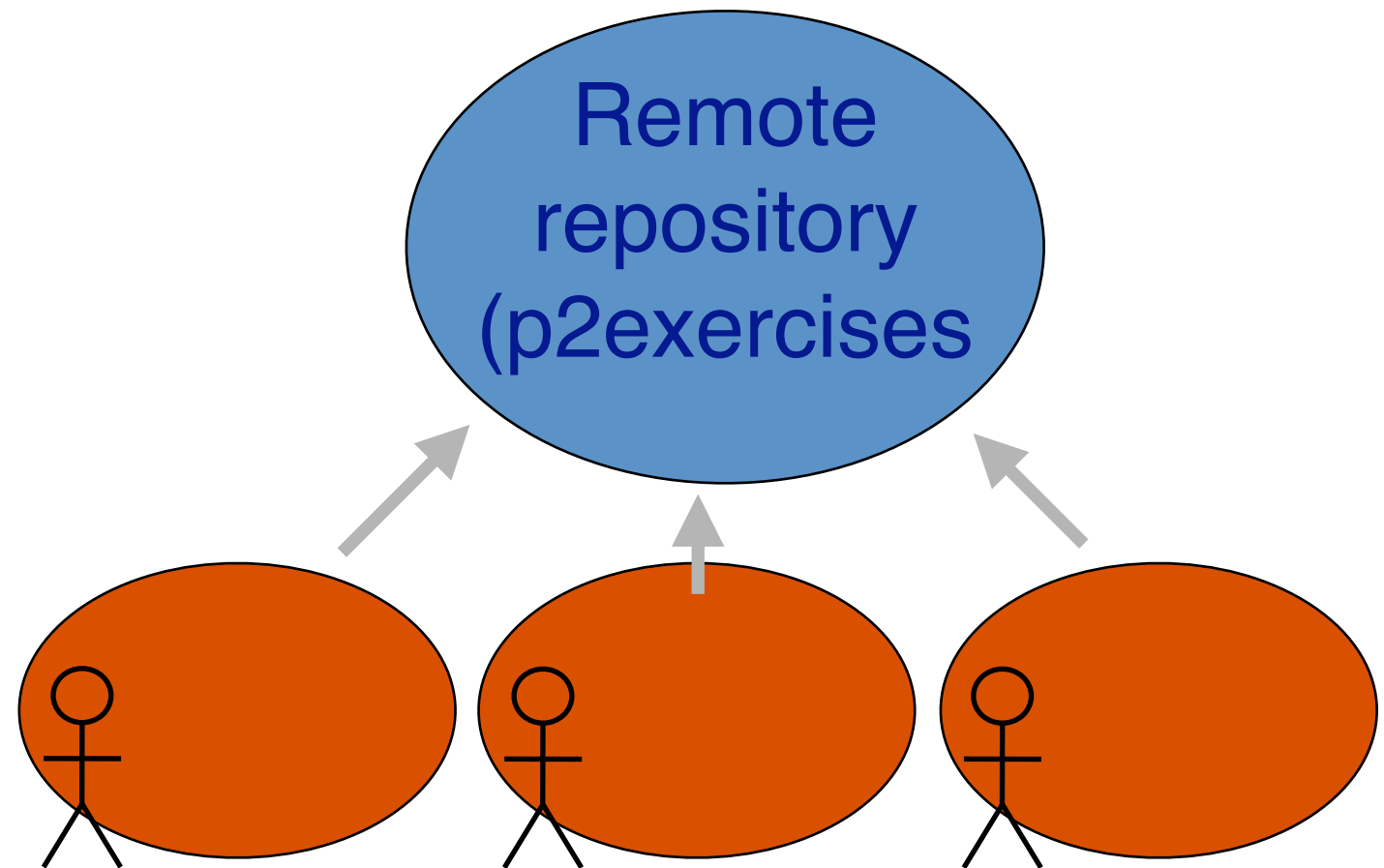
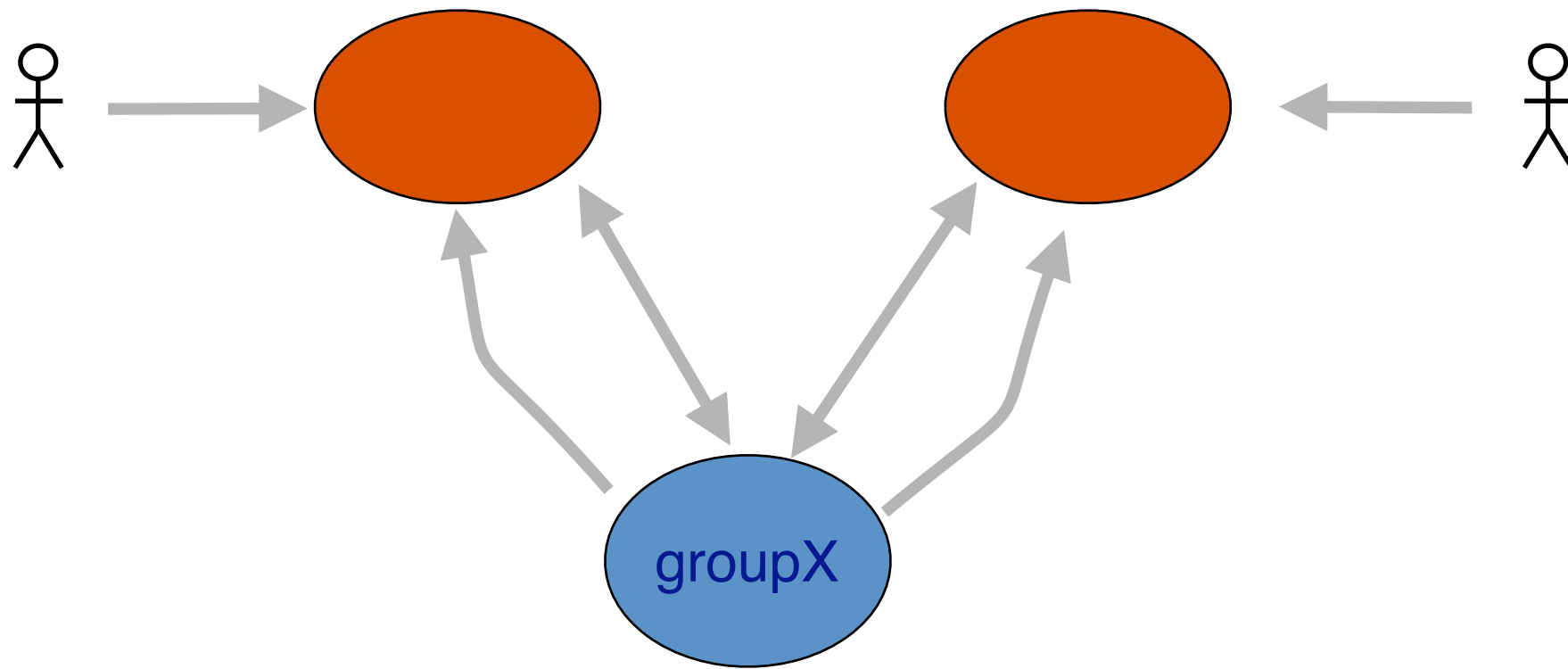


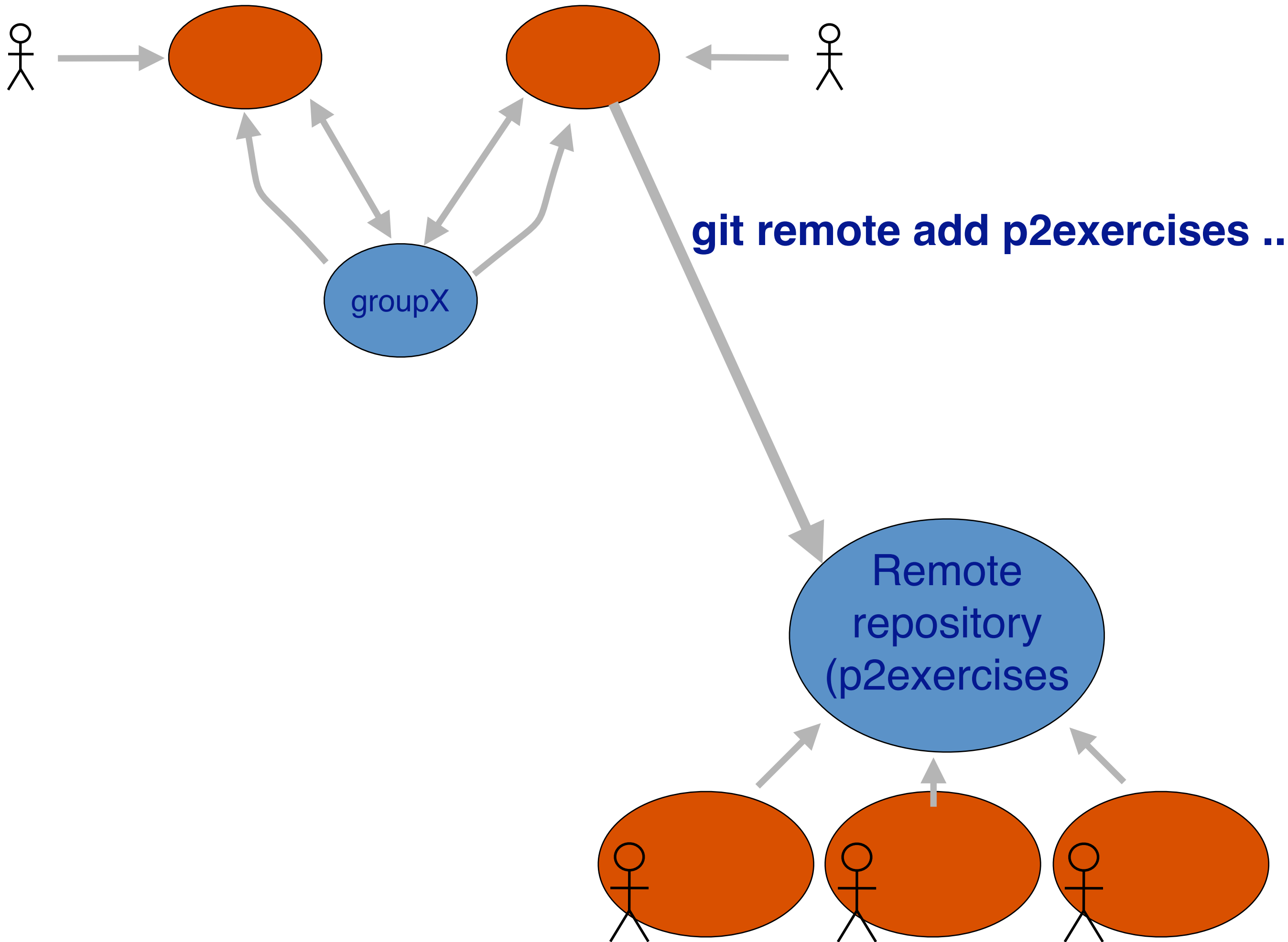
you must pull before every push

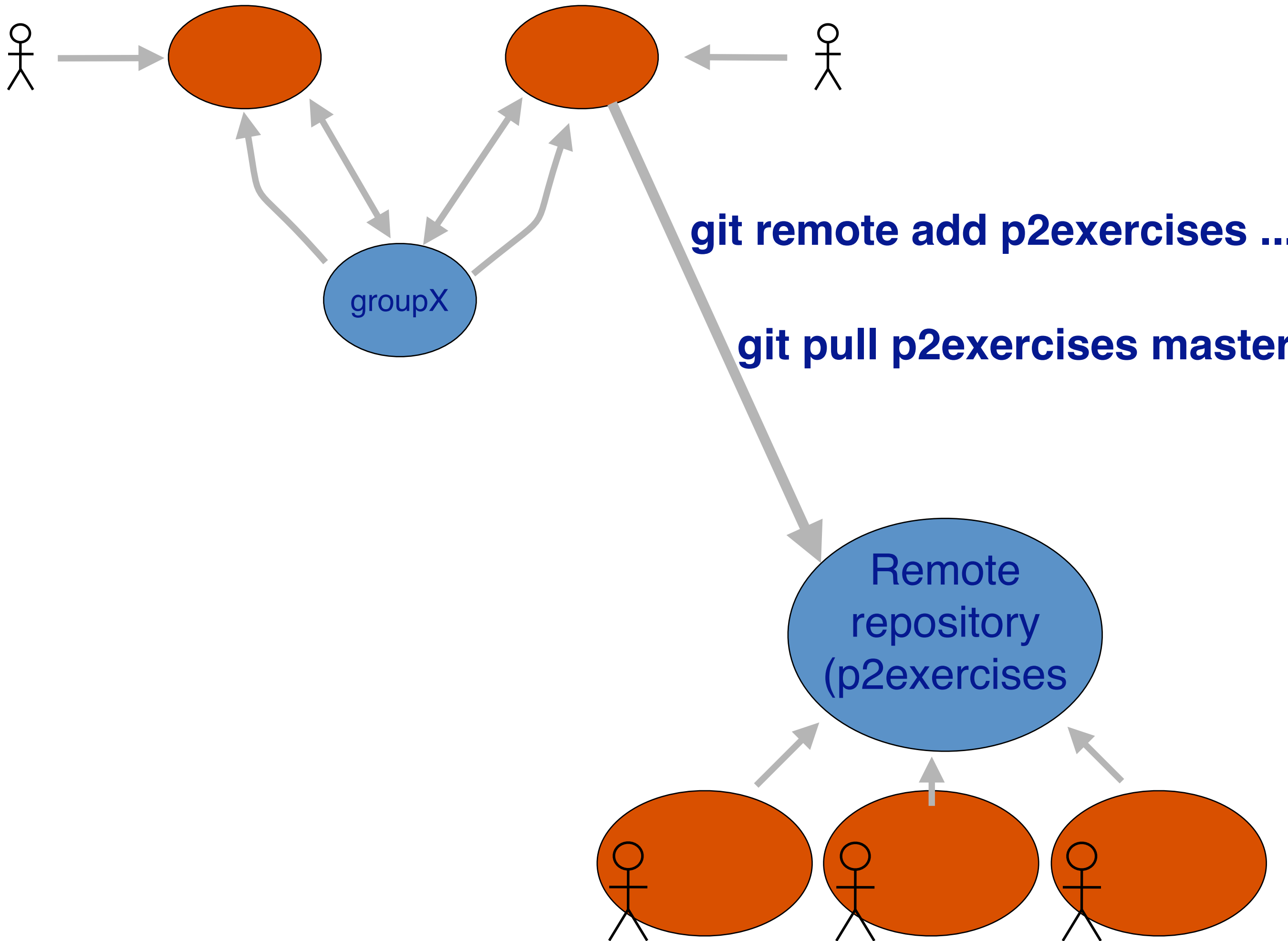


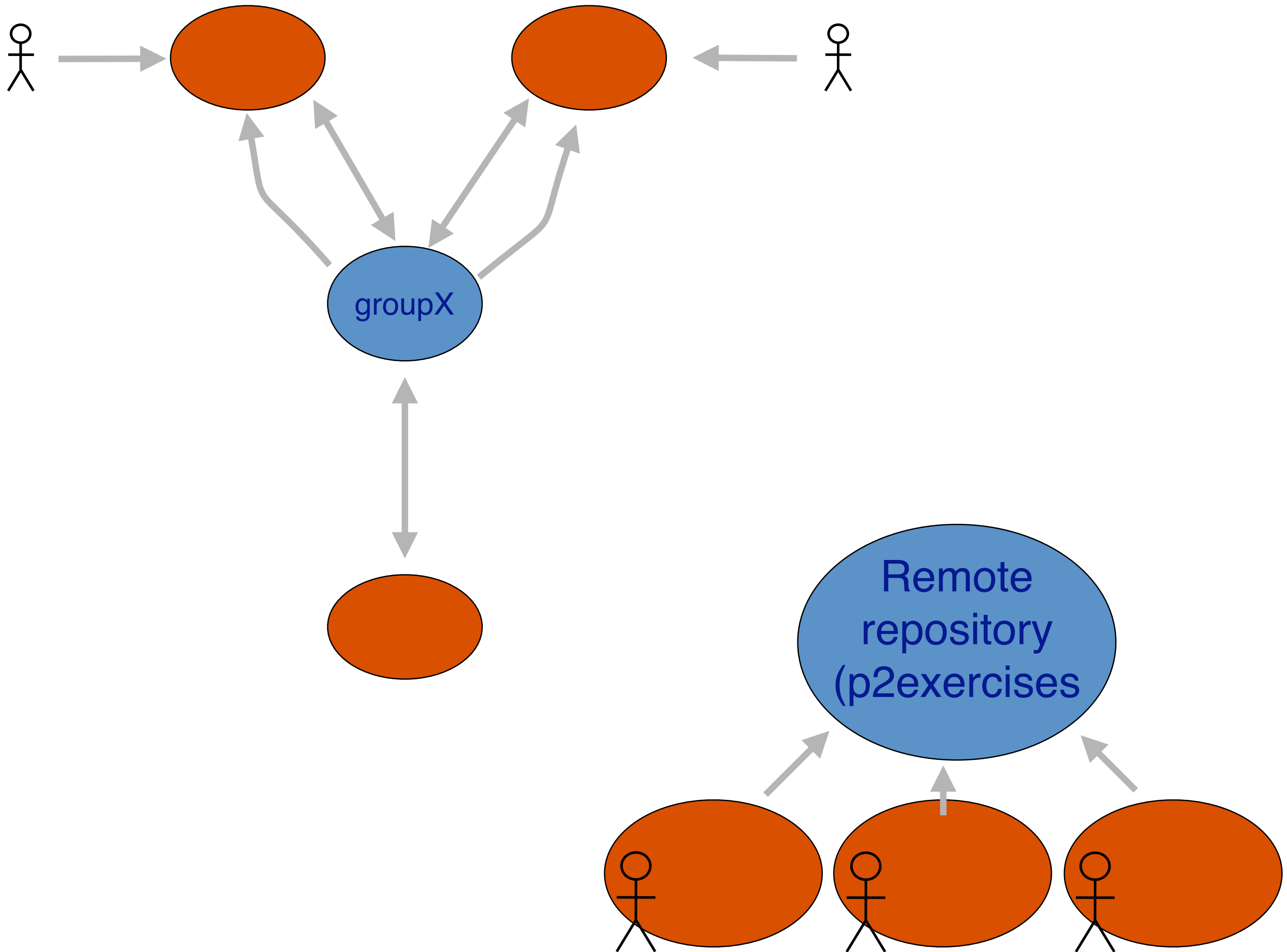


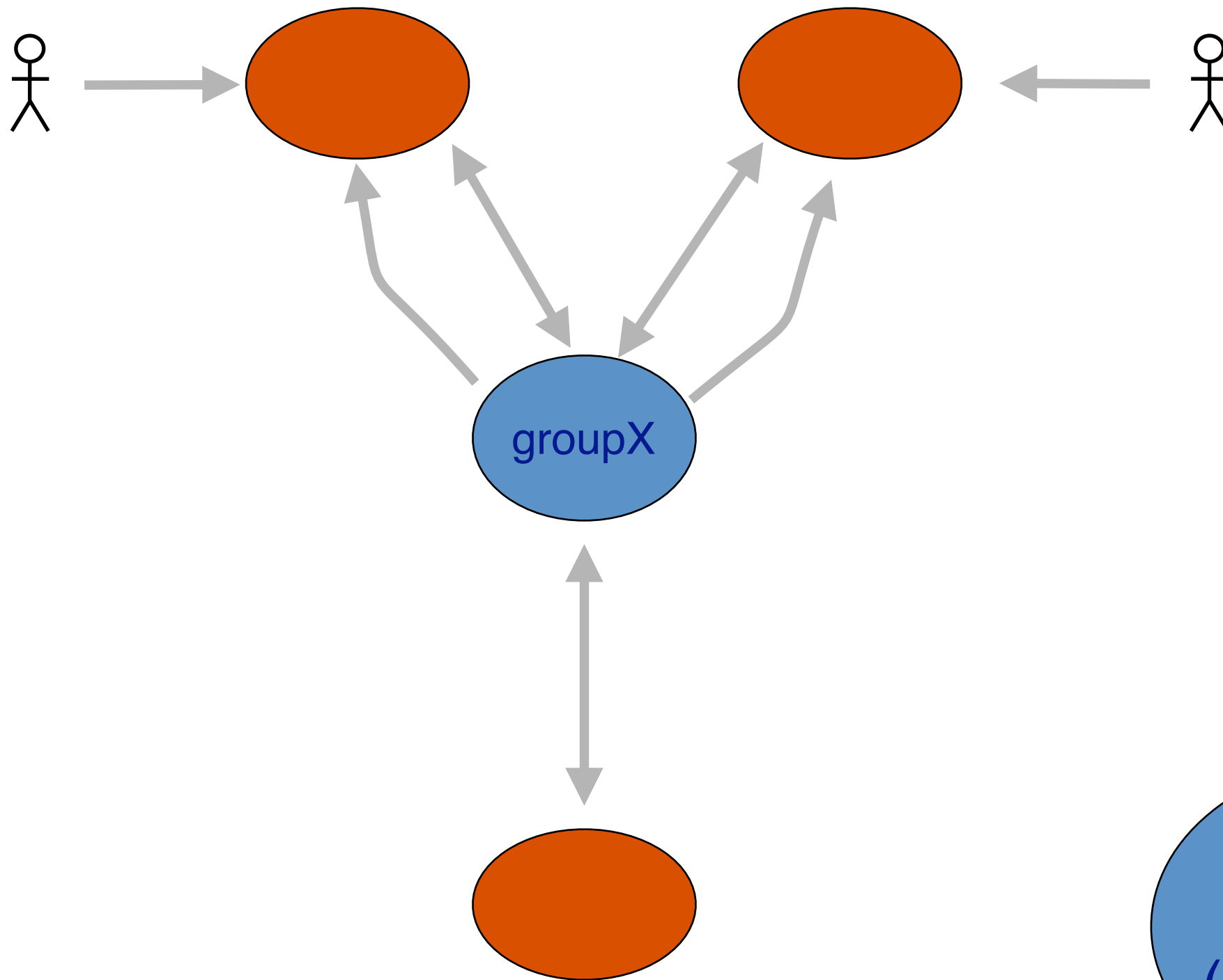
Remote
repository
(p2exercises)



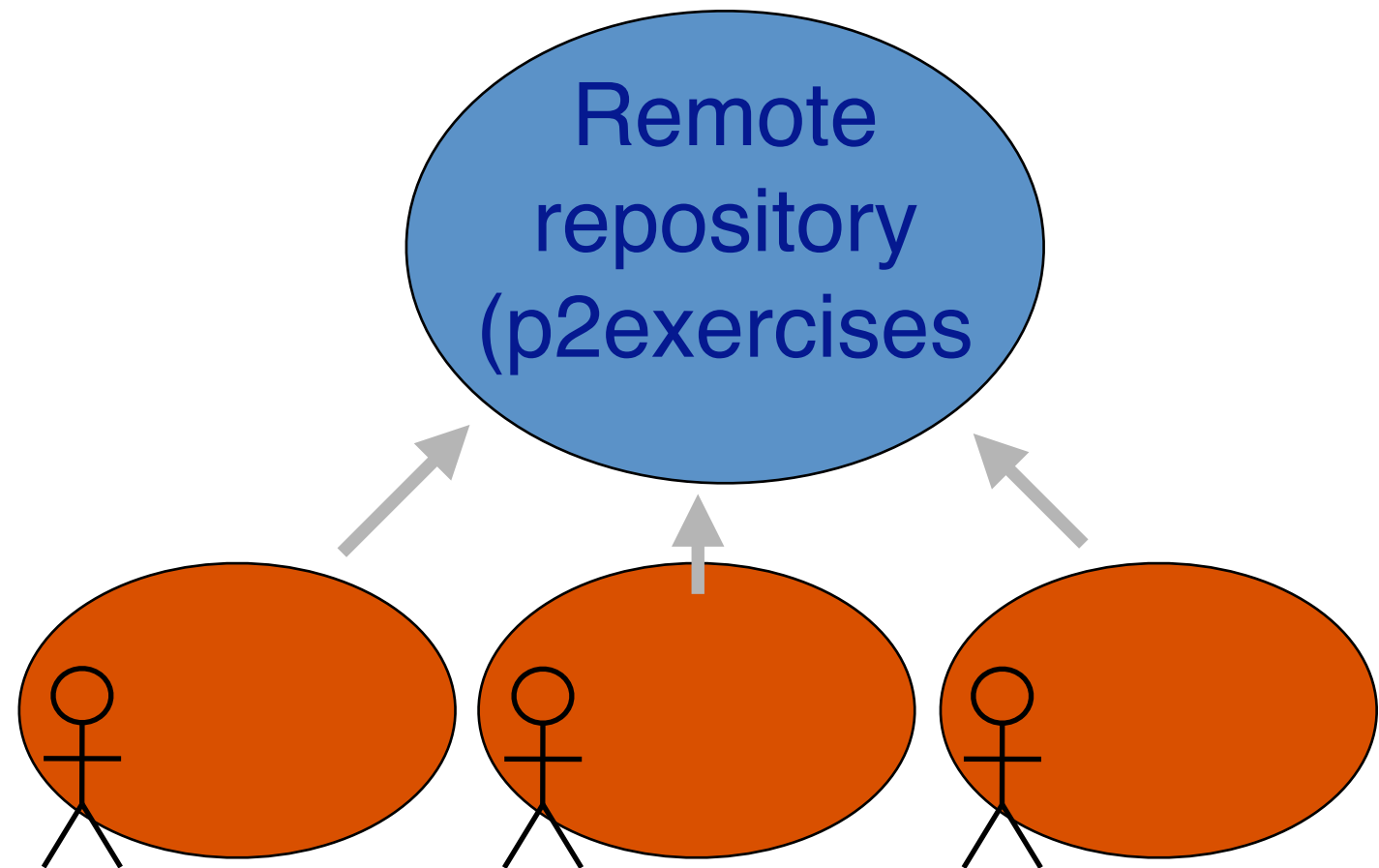








**do not commit after the
deadline; it leads to
merge conflicts**

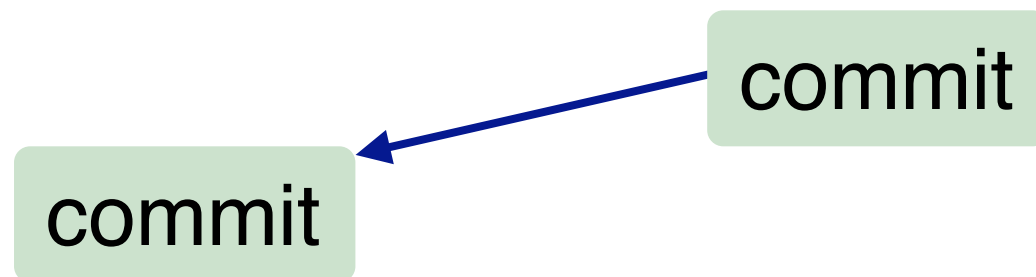


Basic git

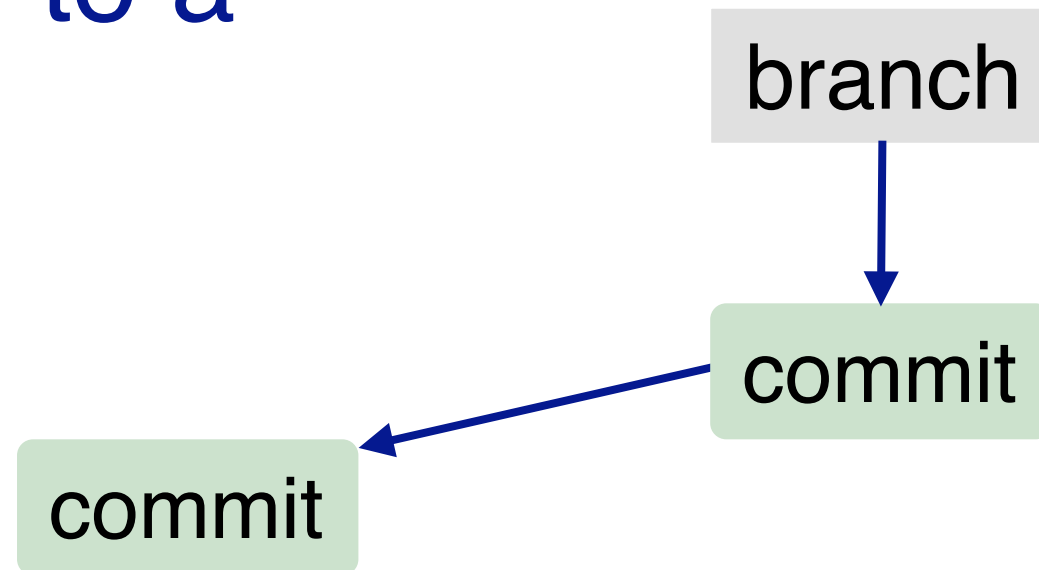
A “commit” is
“a set of changes”
to a “set of files”

commit

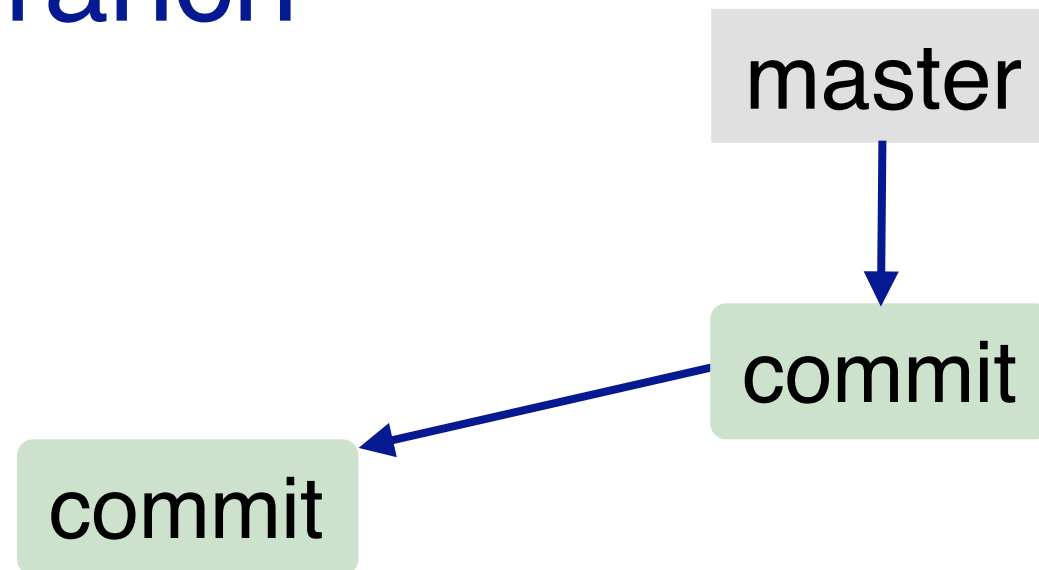
Most commits
modify (or merge)
earlier commits



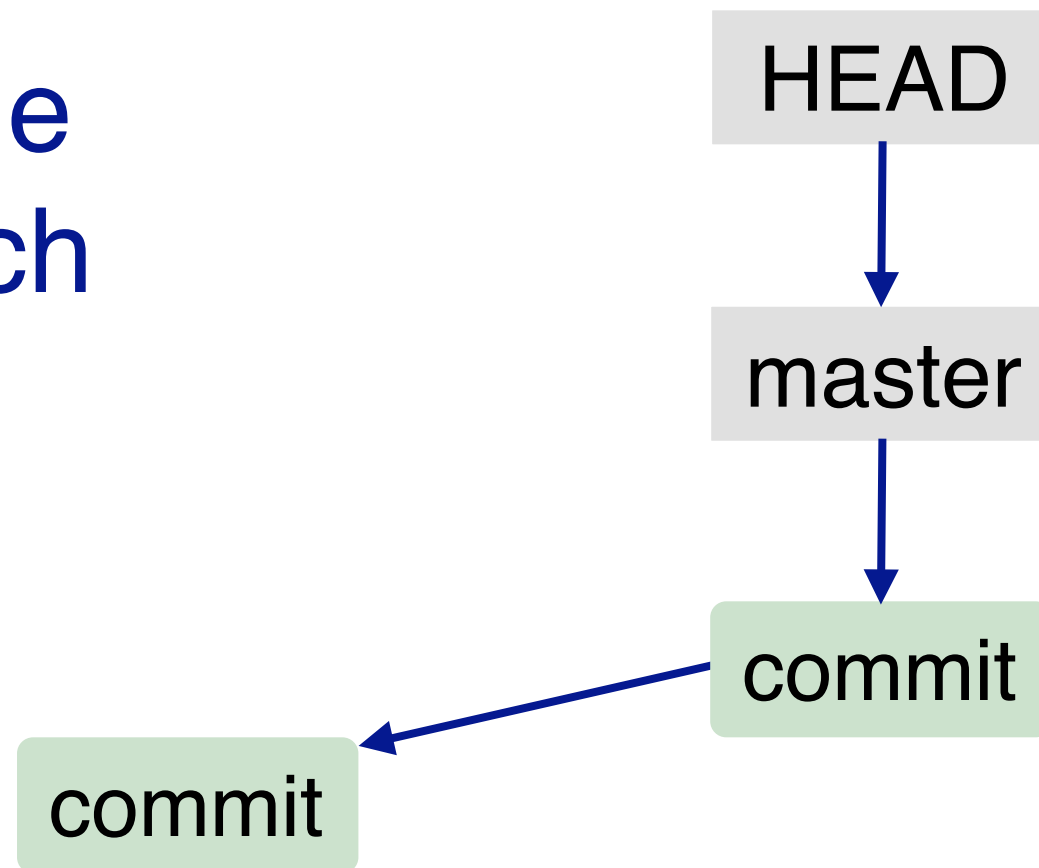
A graph of commits
may belong to a
branch



master
is the main branch

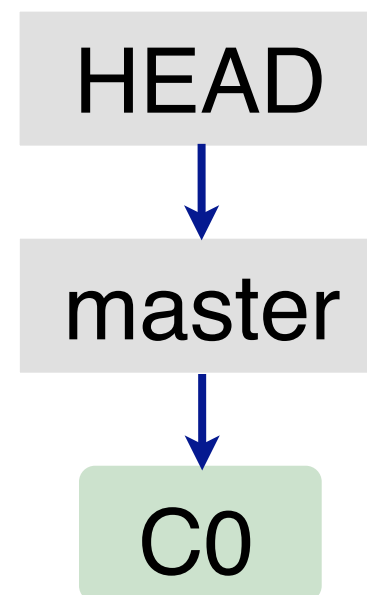


“HEAD “is the
current branch

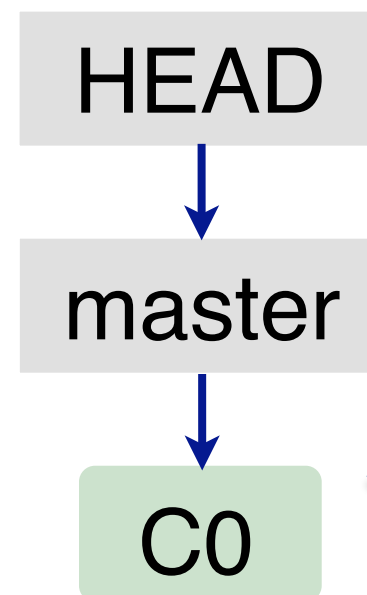


Create a git repo

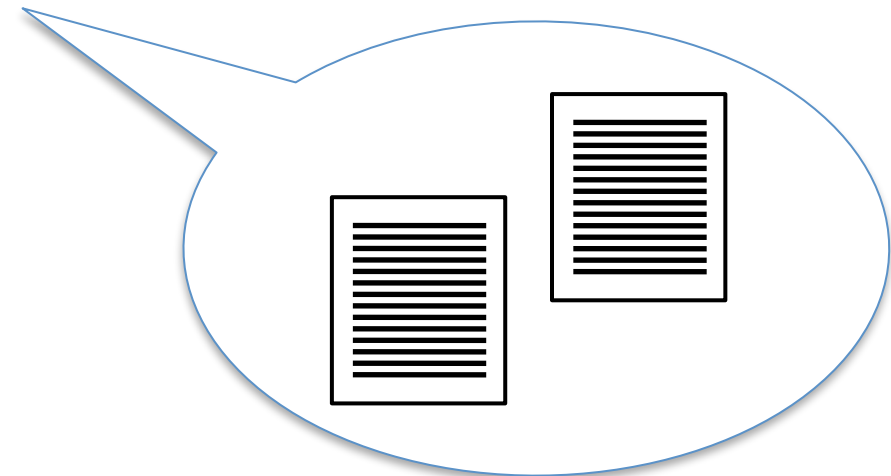
```
mkdir repo  
cd repo  
git init
```

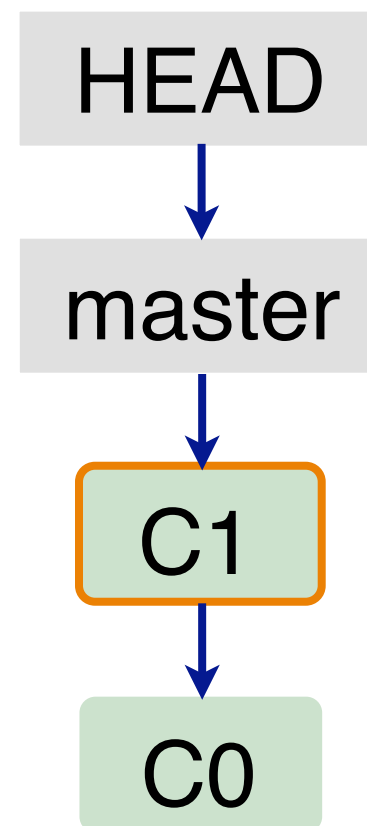


Tell git to “stage”
changes



```
git add ...
```

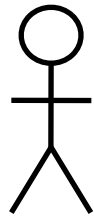




Commit your
changes

```
git commit ...
```


Collaborating

 **John**

Jane 

Local repo

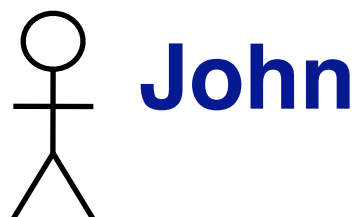
Public repo

Local repo

master

C1

C0



John



Jane

Local repo

git clone ...

master

C1

C0

Public repo

master

C1

C0

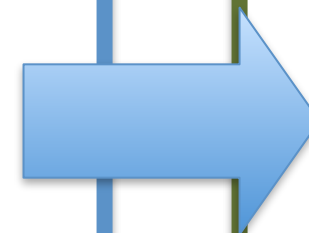
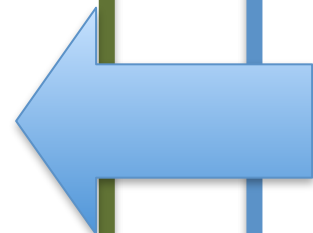
Local repo

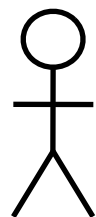
git clone ...

master

C1

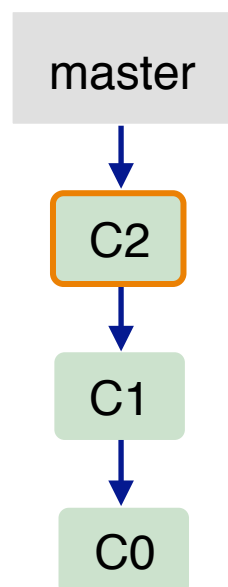
C0



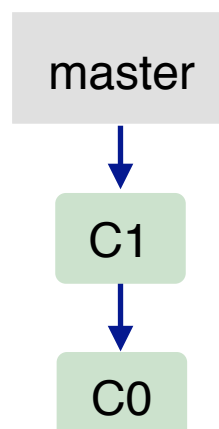
 **John**

Jane 

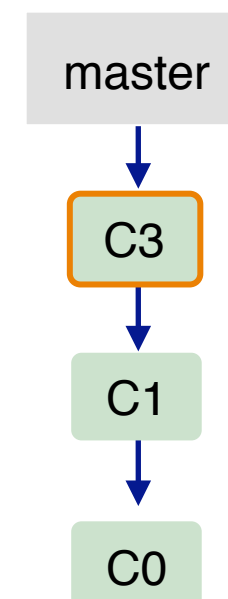
Local repo



Public repo

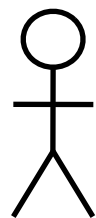


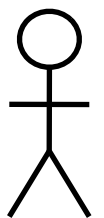
Local repo



git add ...
git commit ...

git add ...
git commit ...

 **John**

Jane 

Local repo

Public repo

Local repo

git pull

master

C2

C1

C0

master

C1

C0

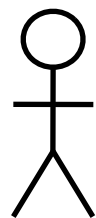
master

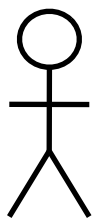
C3

C1

C0

(nothing new to pull)

 **John**

Jane 

Local repo

git push

master

C2

C1

C0

Public repo

master

C2

C1

C0

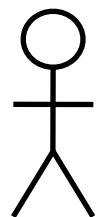
Local repo

master

C3

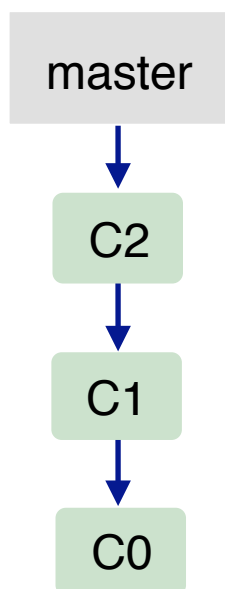
C1

C0

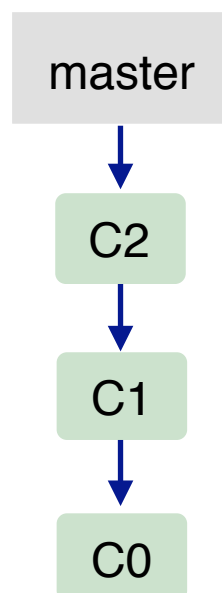
 **John**

Jane 

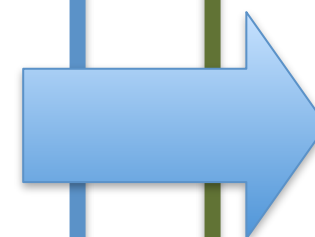
Local repo



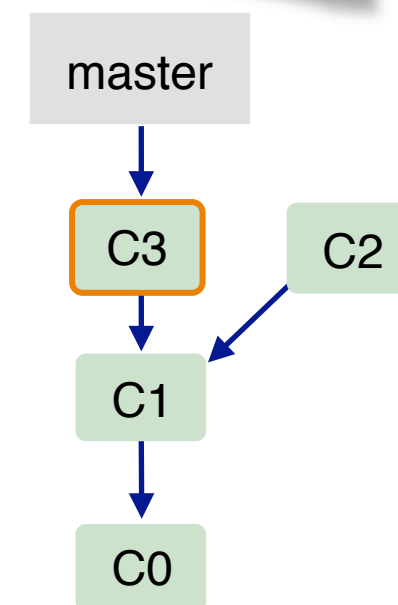
Public repo

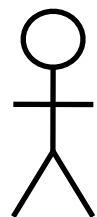


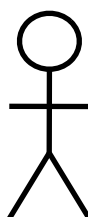
git pull



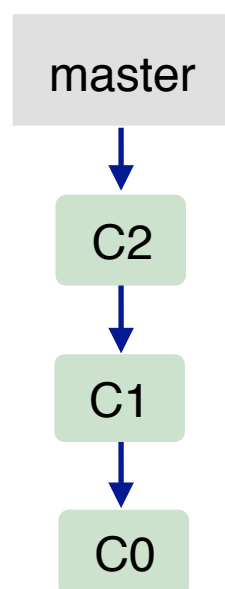
Local repo



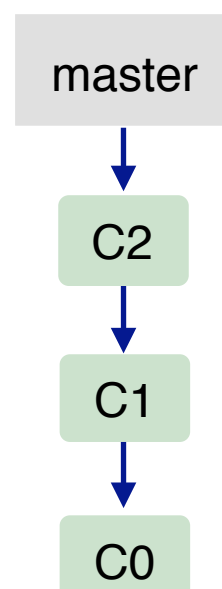
 **John**

Jane 

Local repo

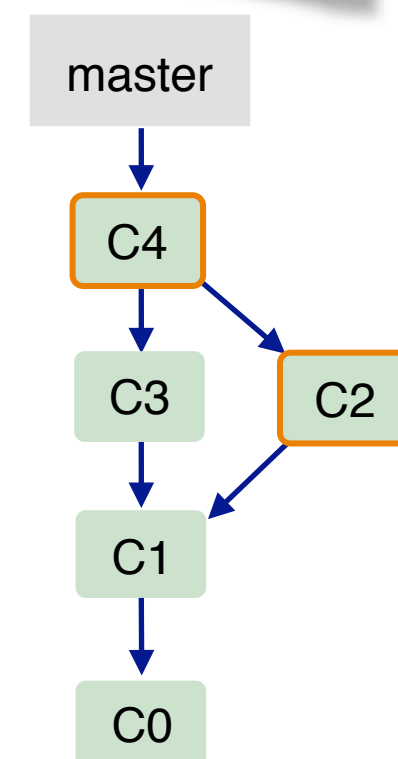


Public repo

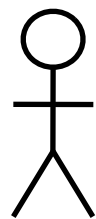


git pull

Local repo

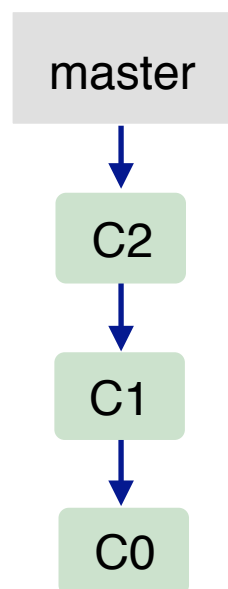


NB: `git pull` = fetch + merge

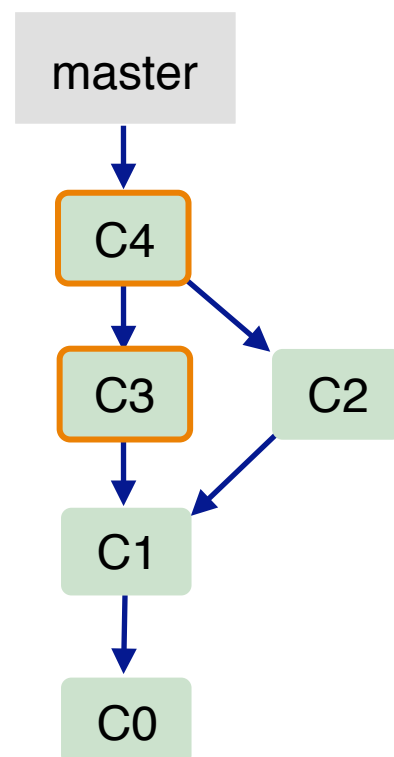
 **John**

Jane 

Local repo

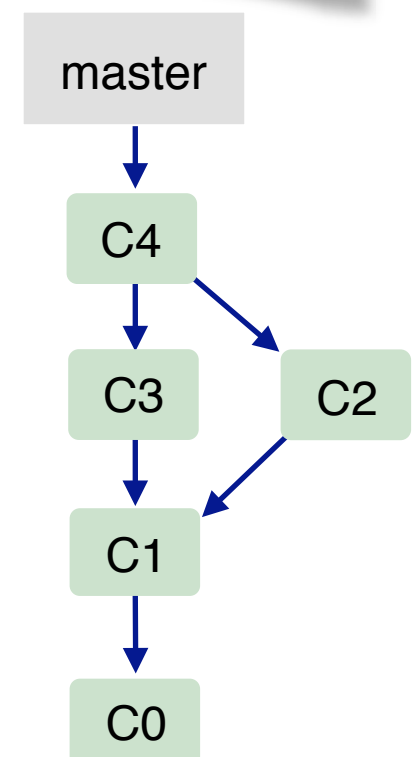


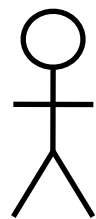
Public repo



git push

Local repo

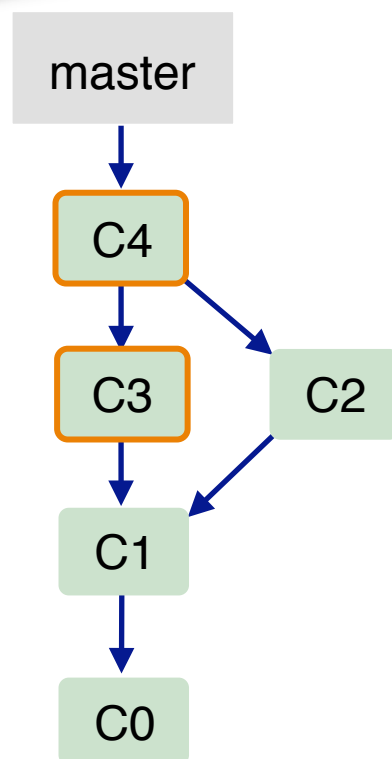


 **John**

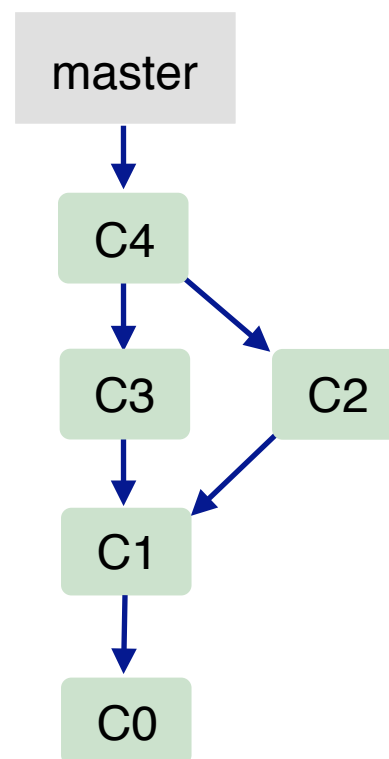
Jane 

Local repo

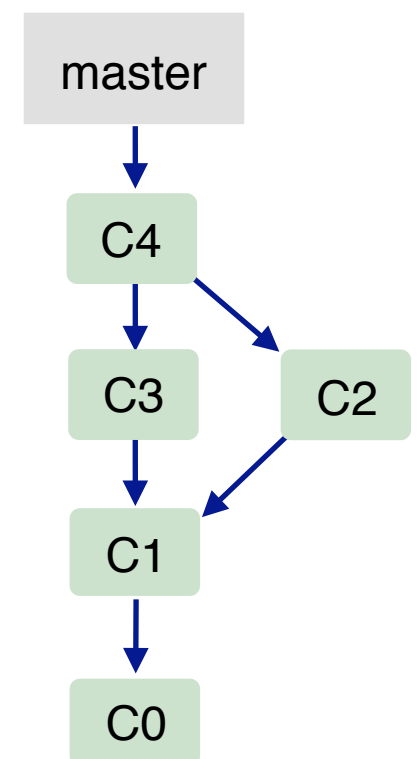
git pull



Public repo



Local repo



to be continued

Resources



<http://git-scm.com/>



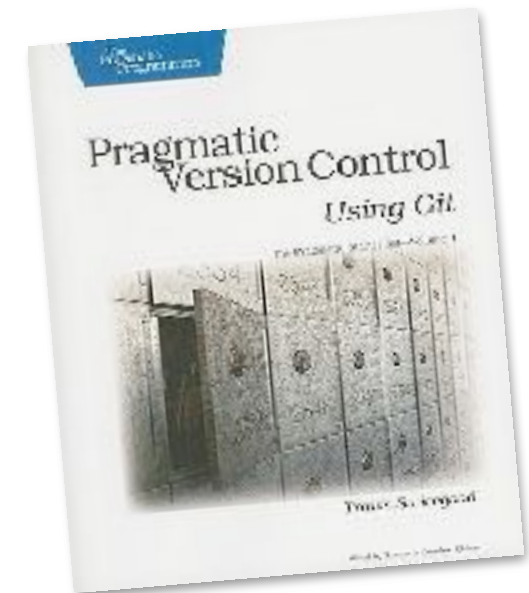
<http://book.git-scm.com/index.html>



<https://github.com/>

Getting Git
Scott Chacon

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



<http://oreilly.com/>



Attribution-ShareAlike 3.0

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions:



Attribution. You must attribute the work in the manner specified by the author or licensor.



Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

<http://creativecommons.org/licenses/by-sa/3.0/>