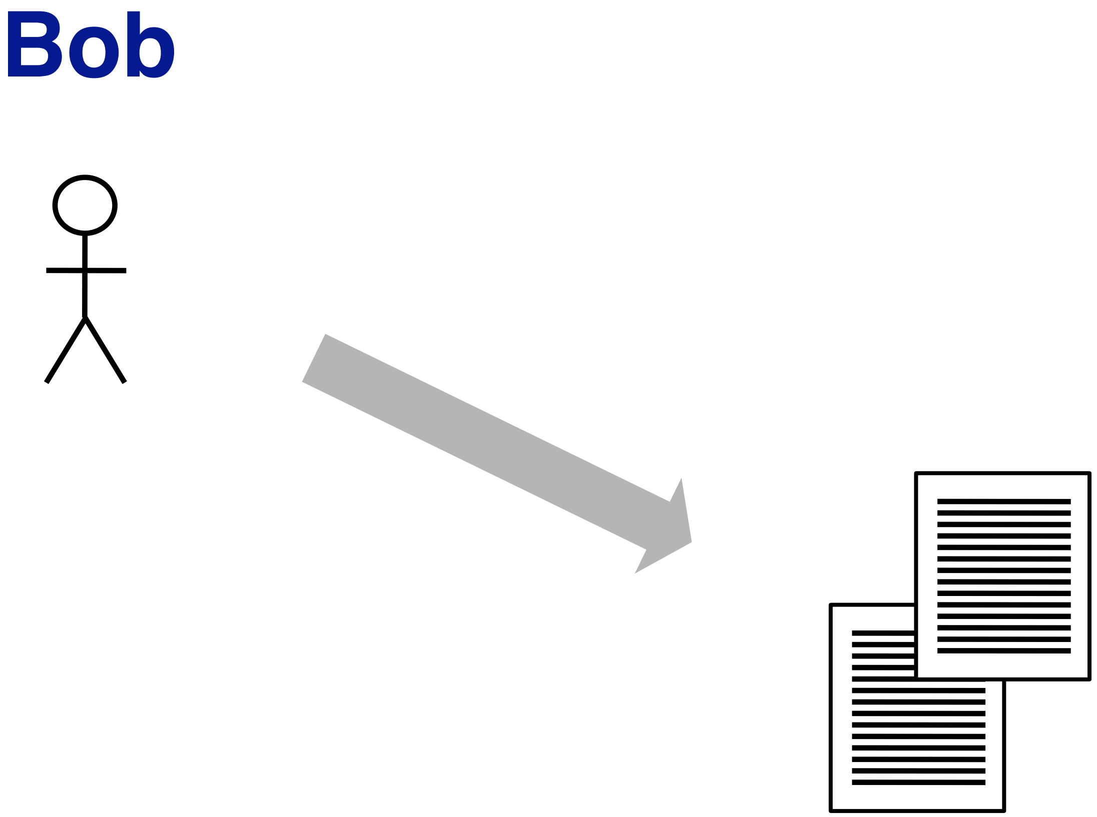
Distributed version control with git — a brief introduction

Andrei Chis

based on slides by
Oscar Nierstrasz

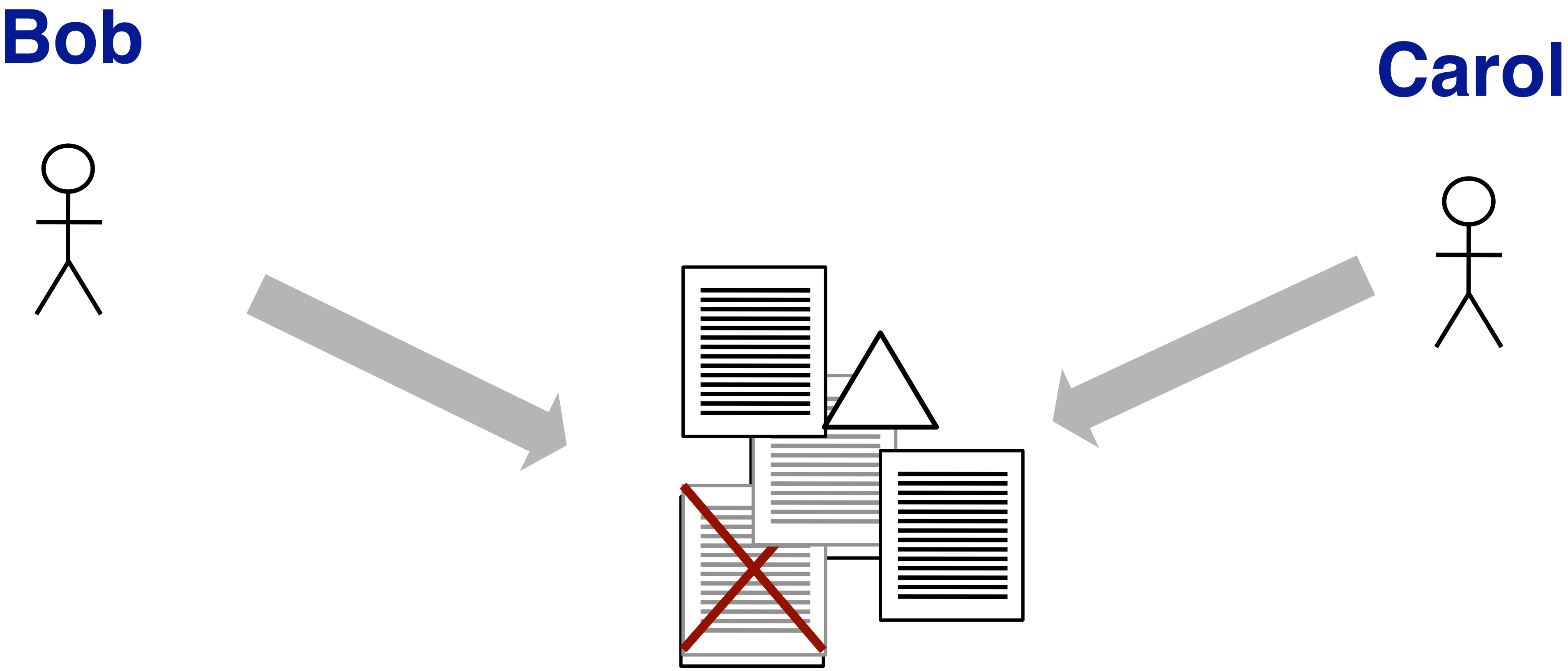
Why git?

Why git?



Bob

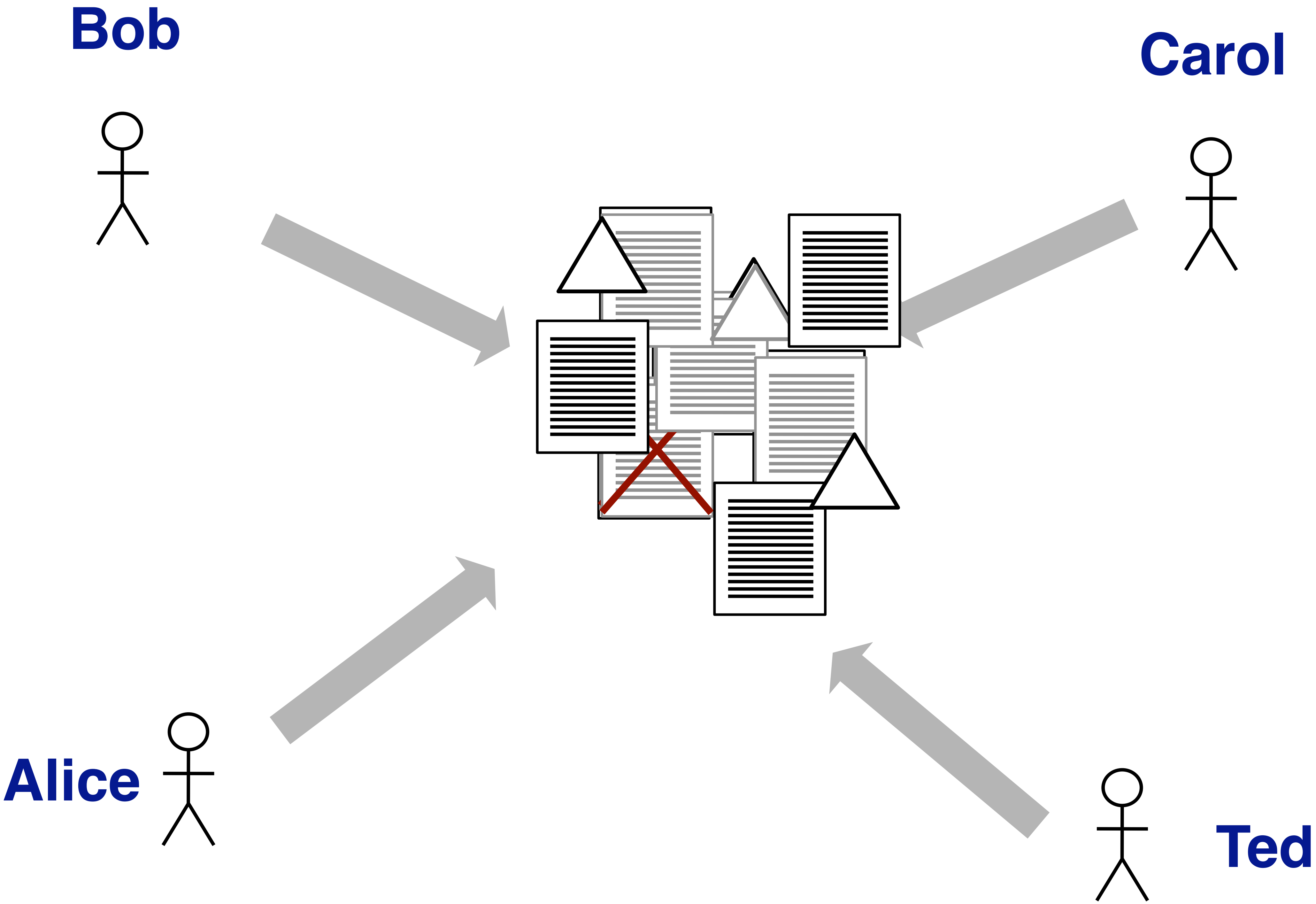
Why git?



Bob

Carol

Why git?



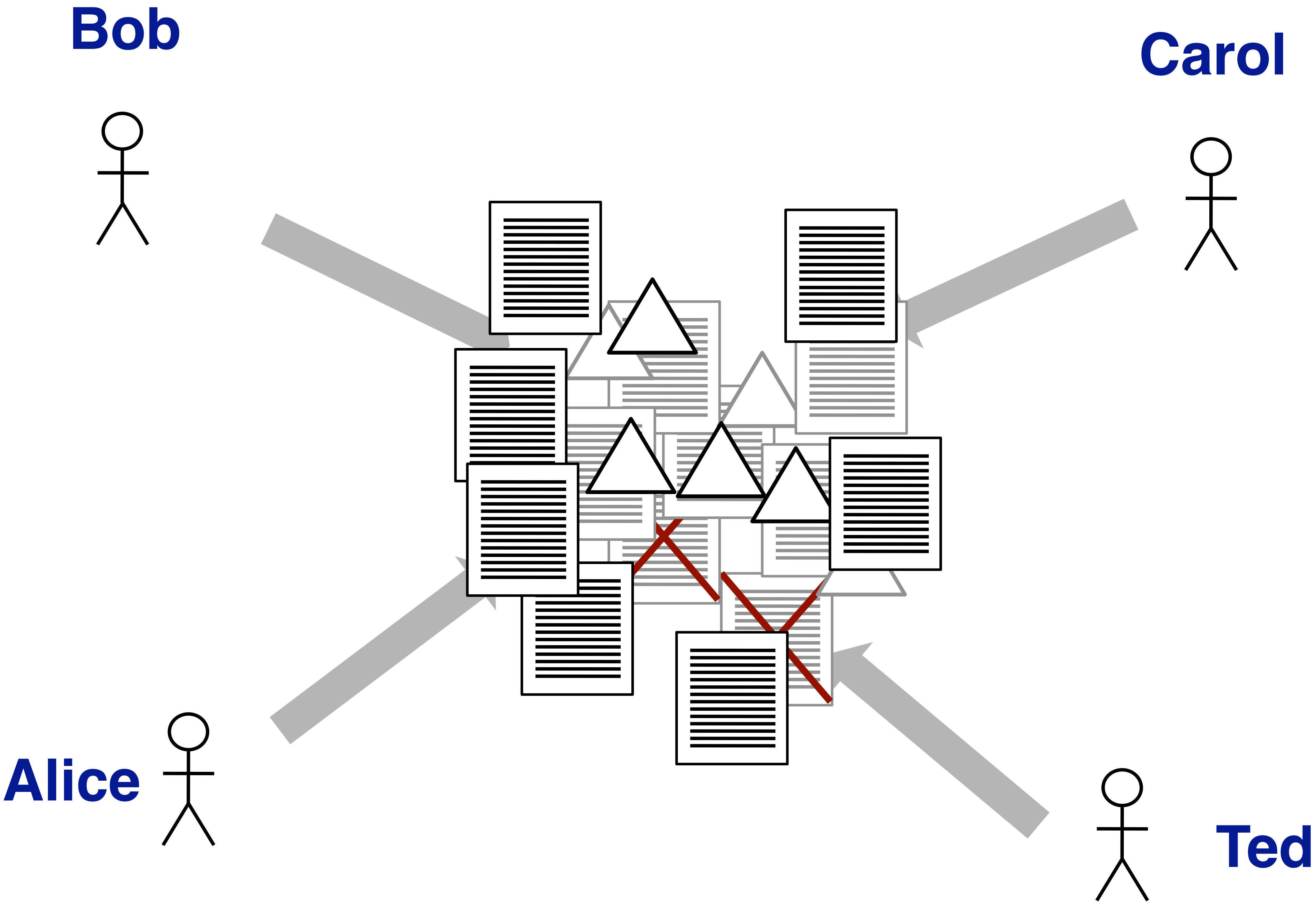
Bob

Carol

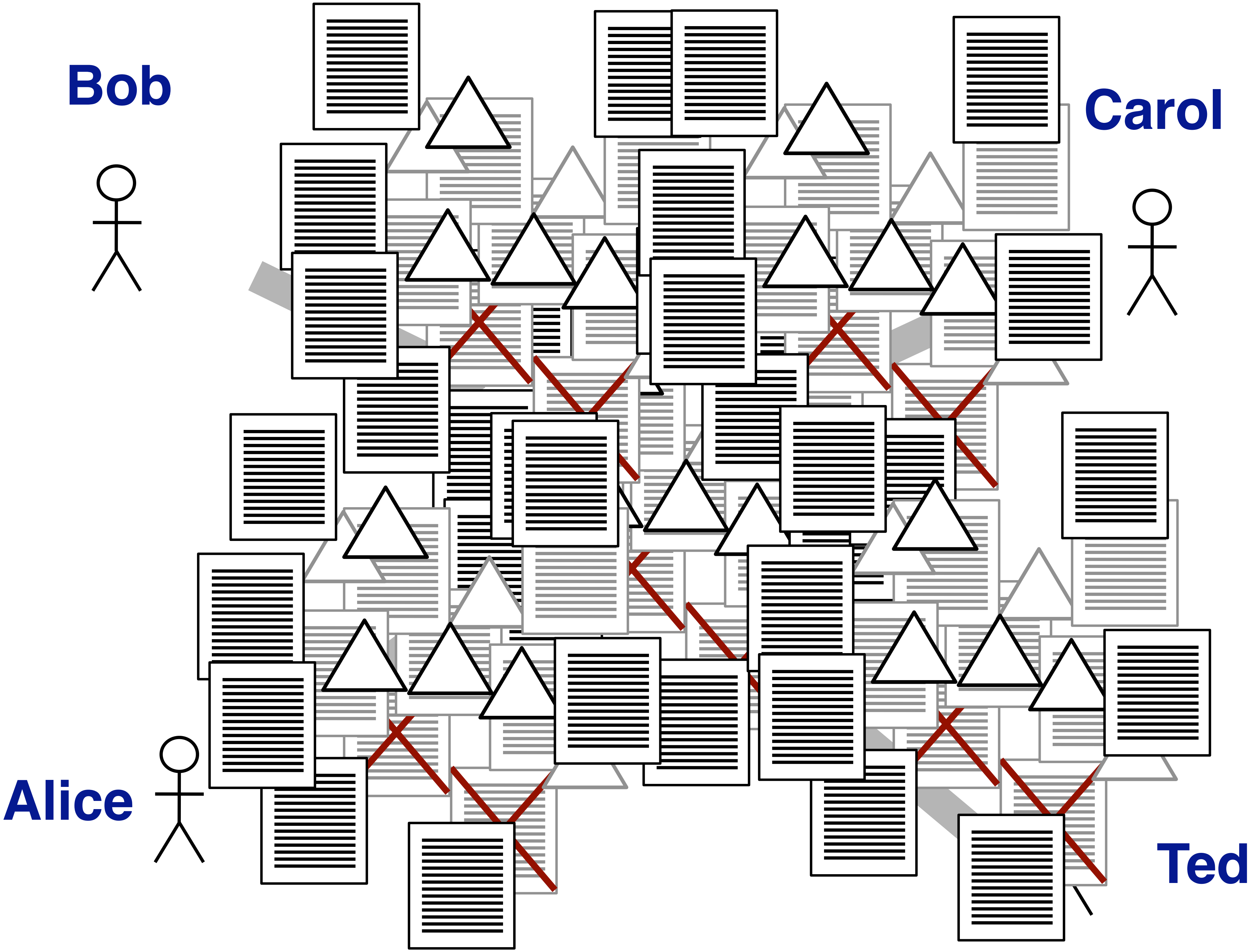
Alice

Ted

Why git?



Why git?

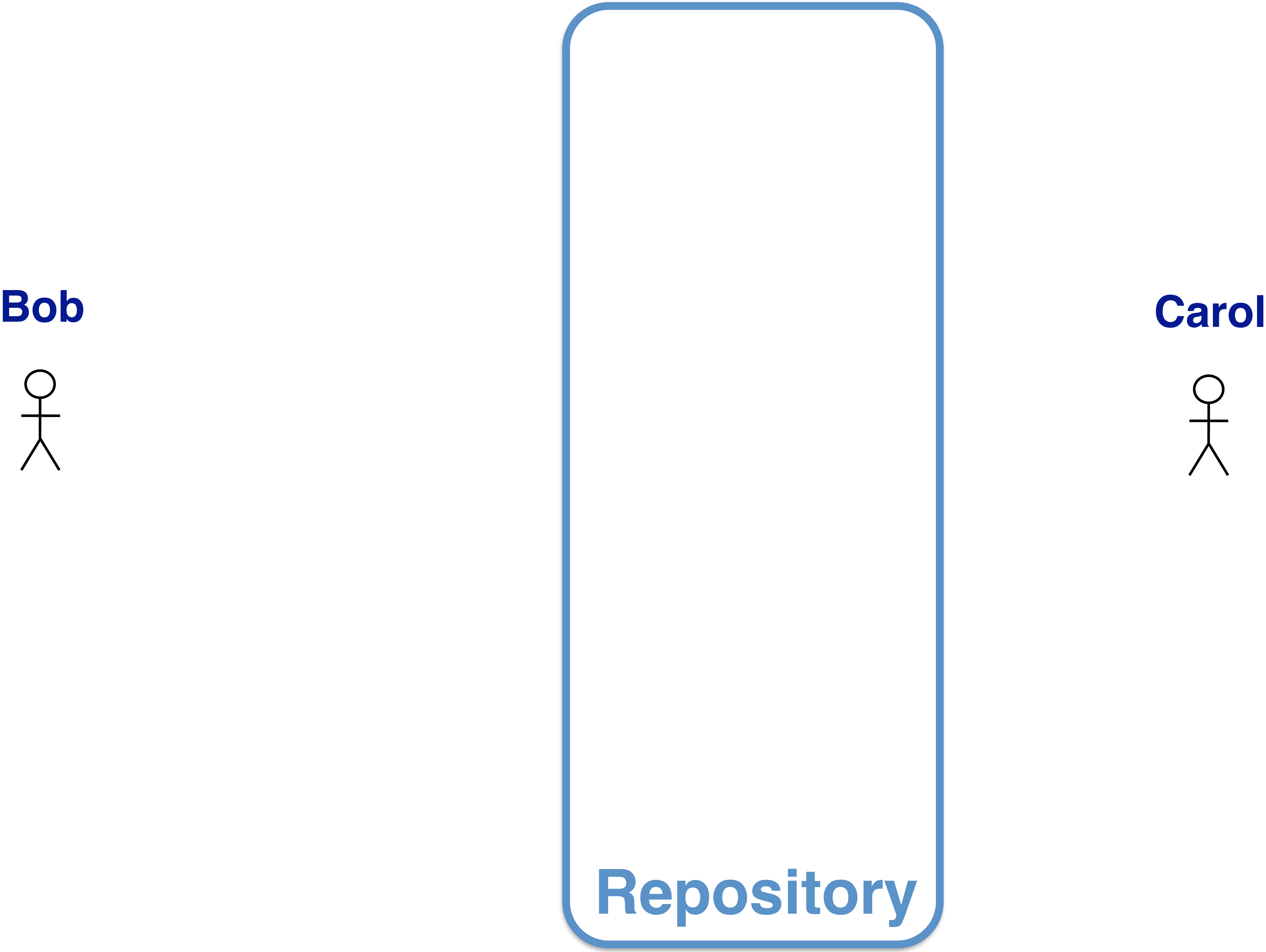


Why git?

A recipe for disaster!

Why git?

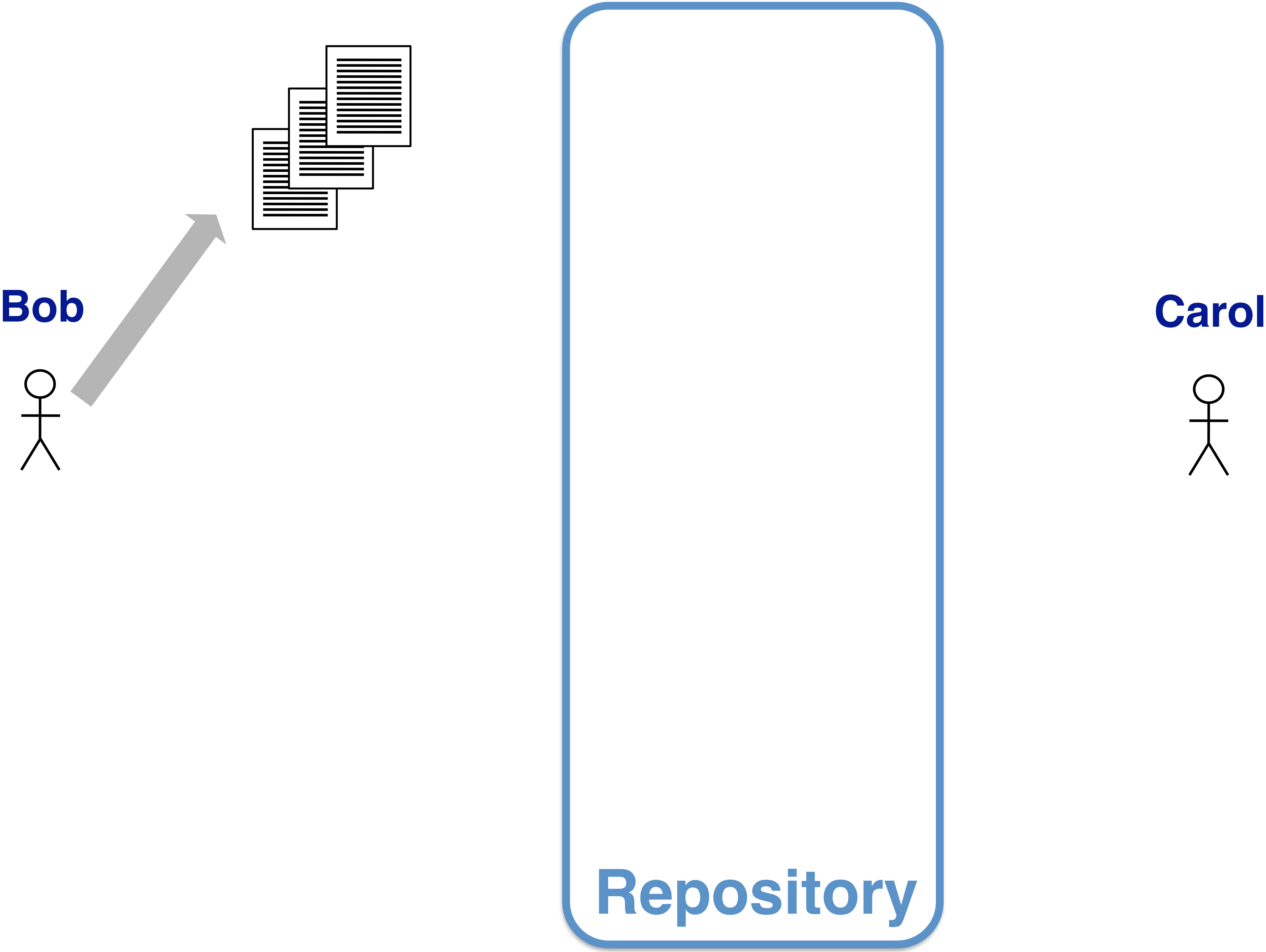
Cope with the confusion that happens when multiple people edit the same files



Bob

Repository

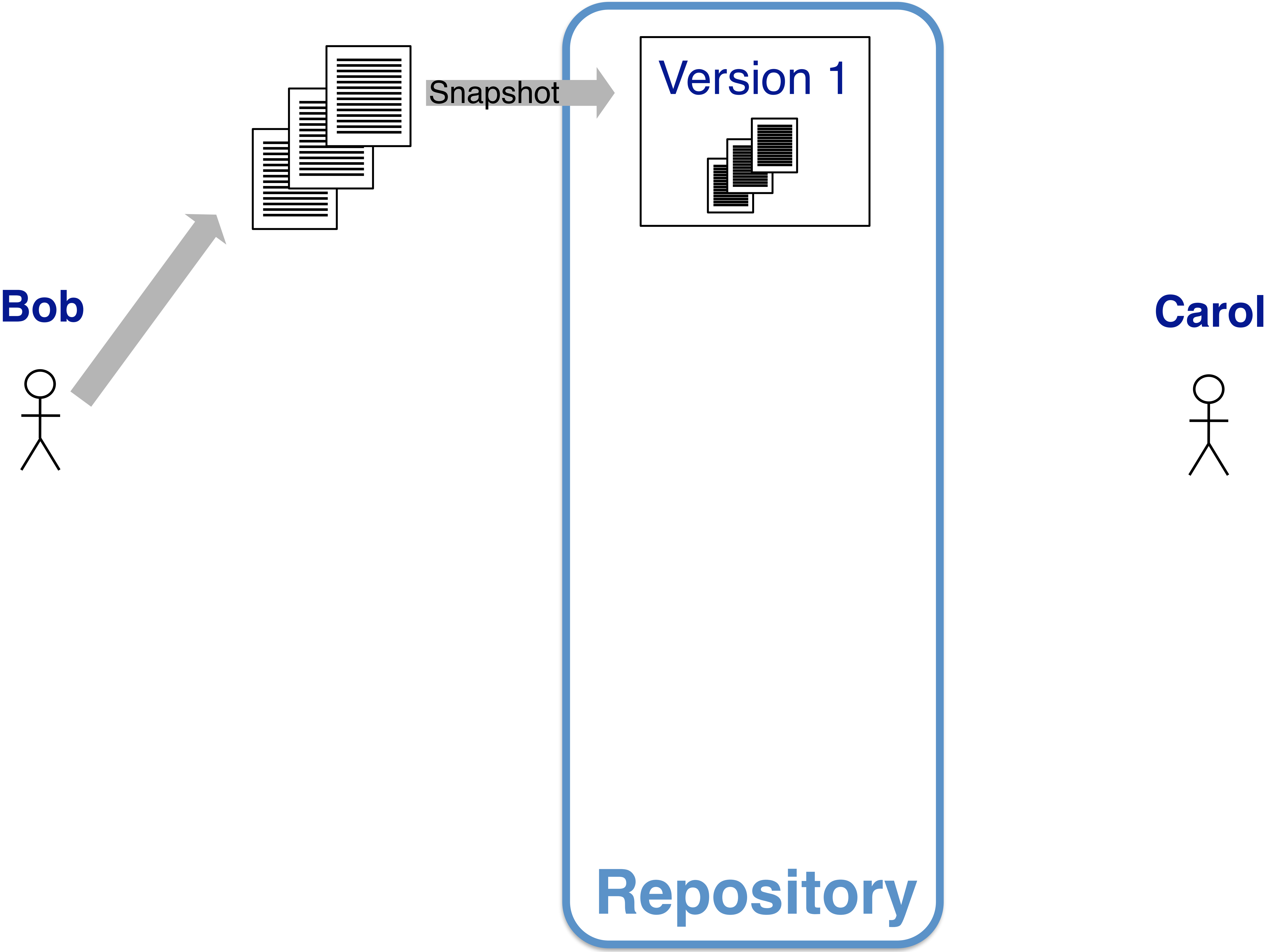
Carol



Bob

Carol

Repository



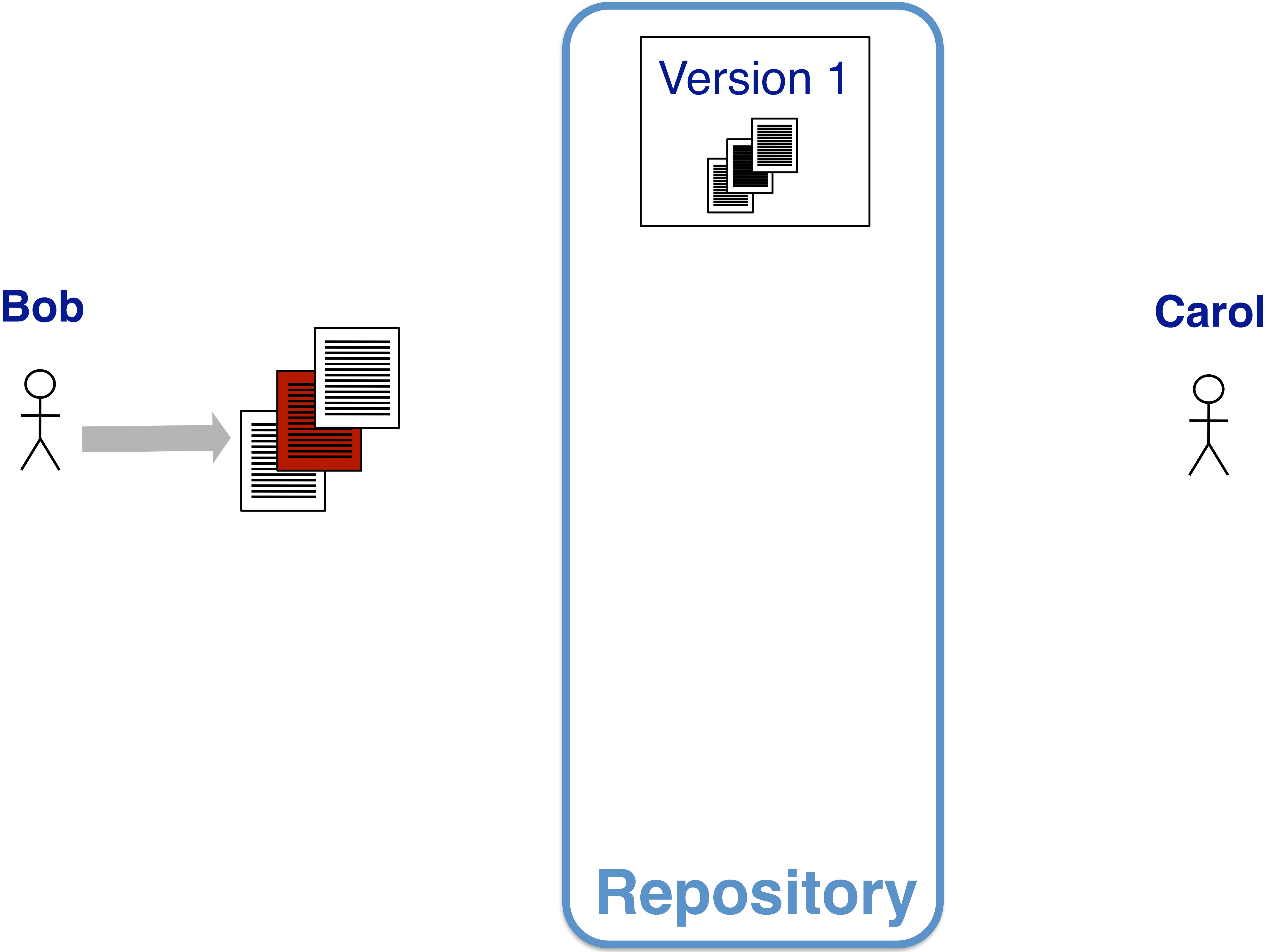
Bob

Snapshot

Version 1

Repository

Carol

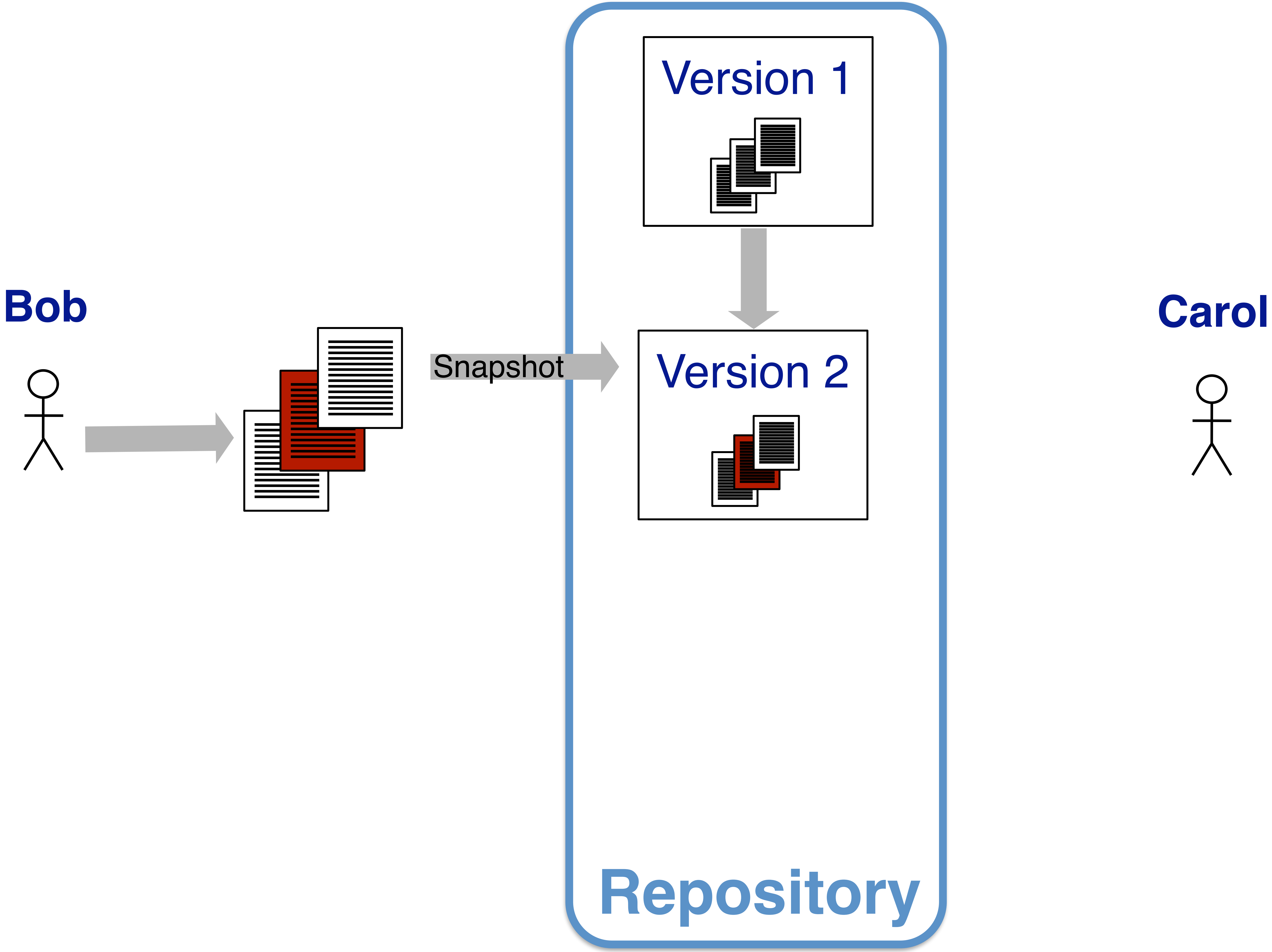


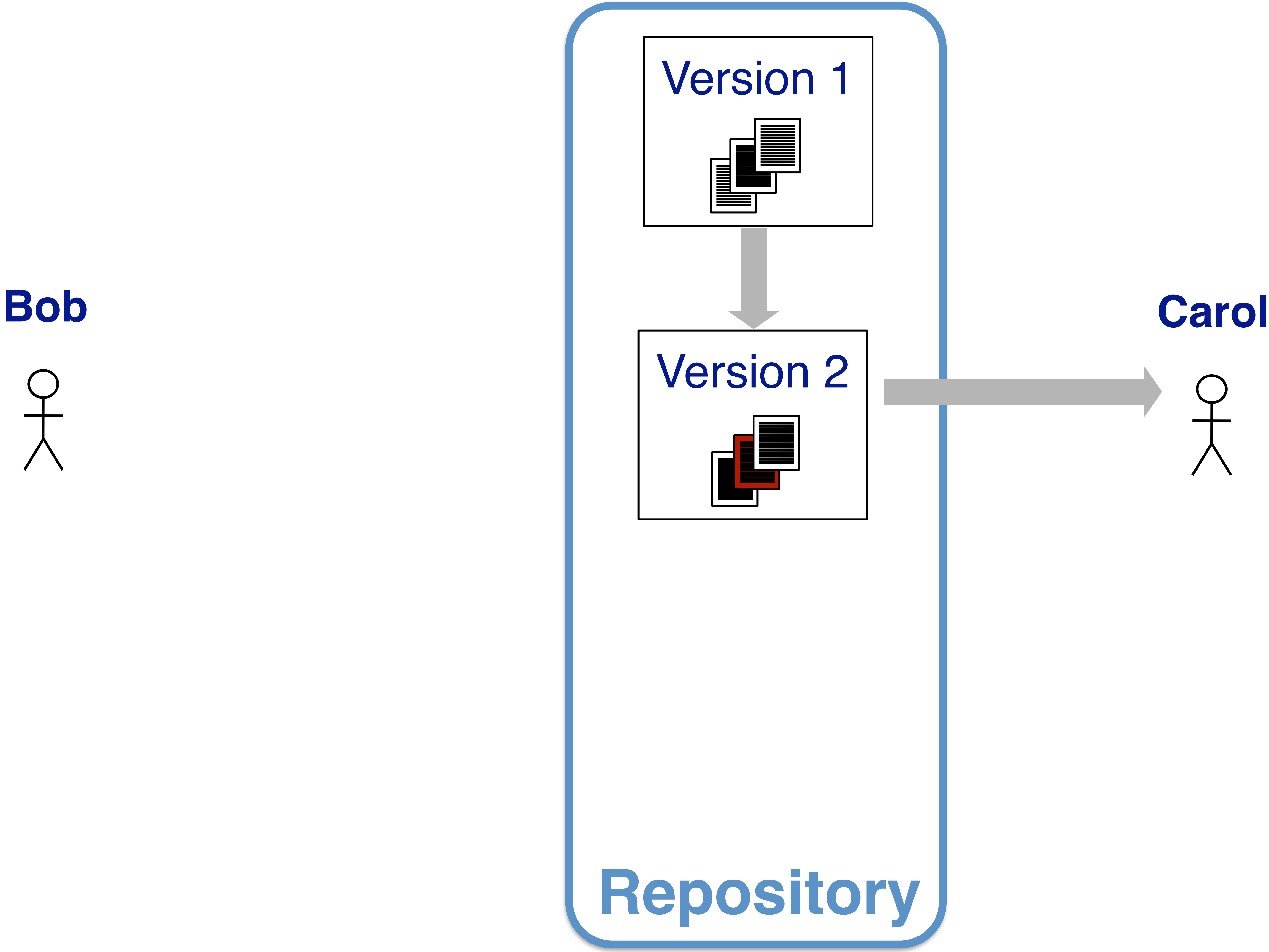
Version 1

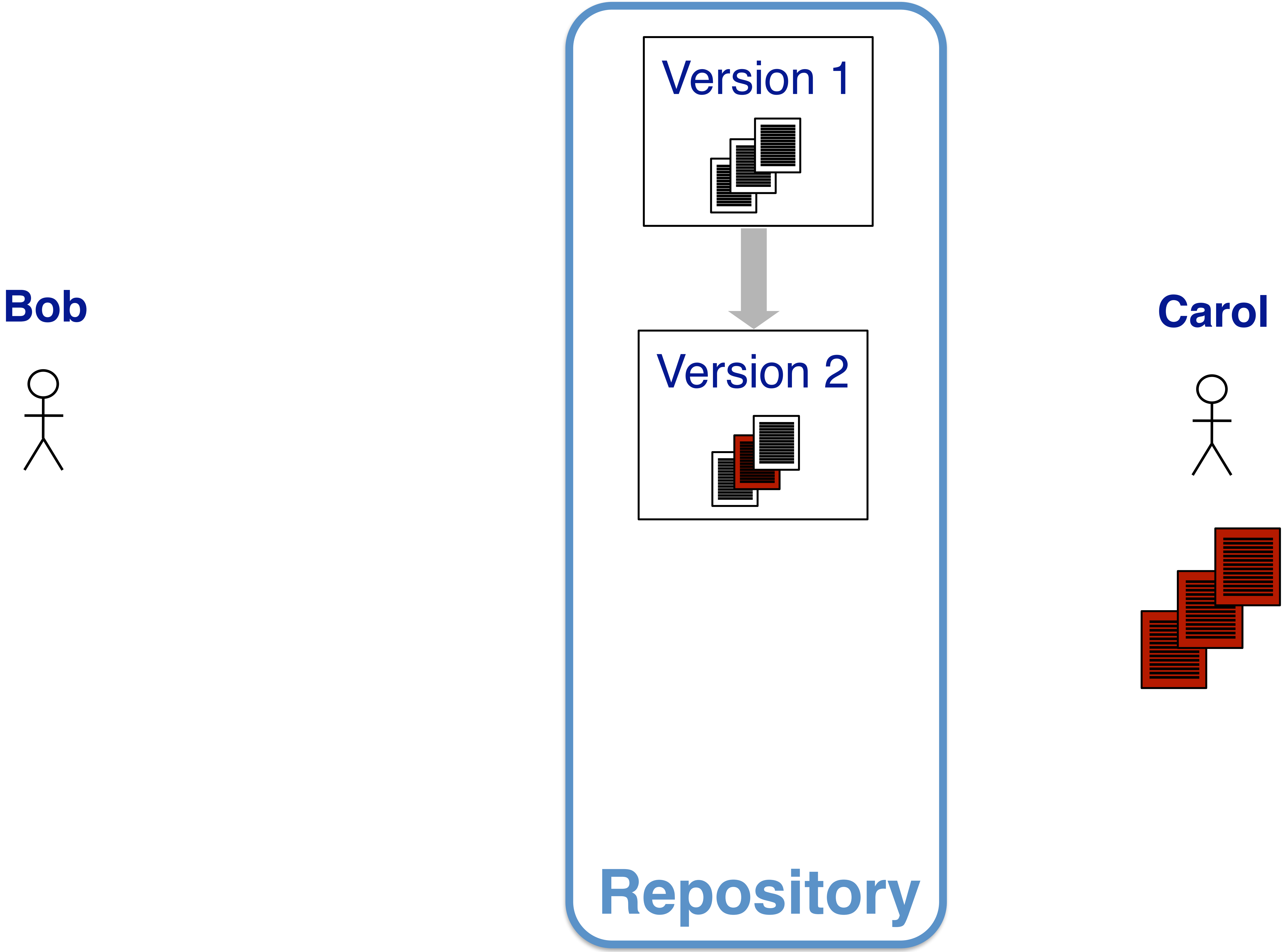
Bob

Carol

Repository


Bob
Snapshot
Version 1
Version 2
Repository
Carol


Version 1
Version 2
Bob
Carol
Repository



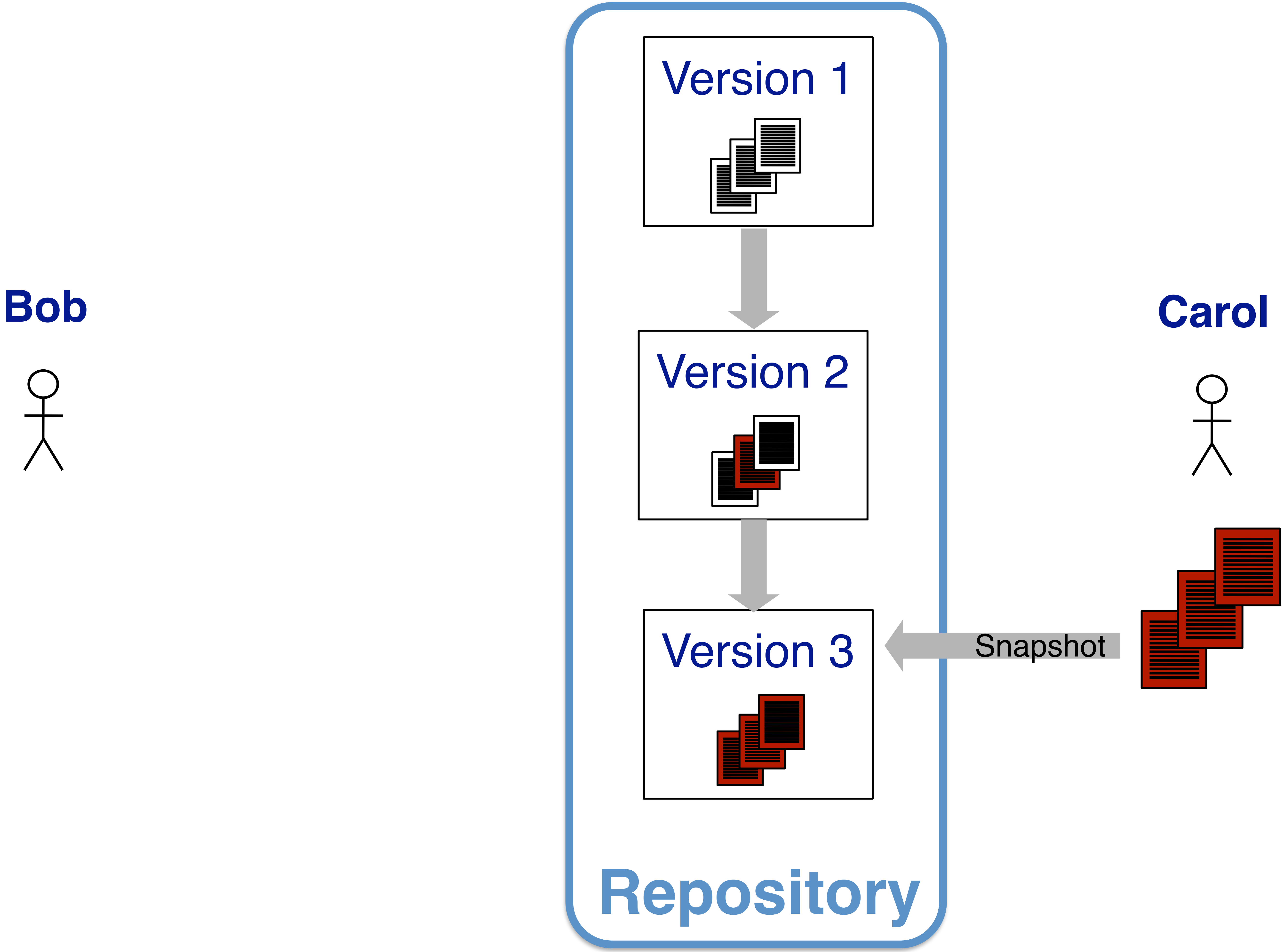
Bob

Version 1

Version 2

Repository

Carol



Bob

Version 1

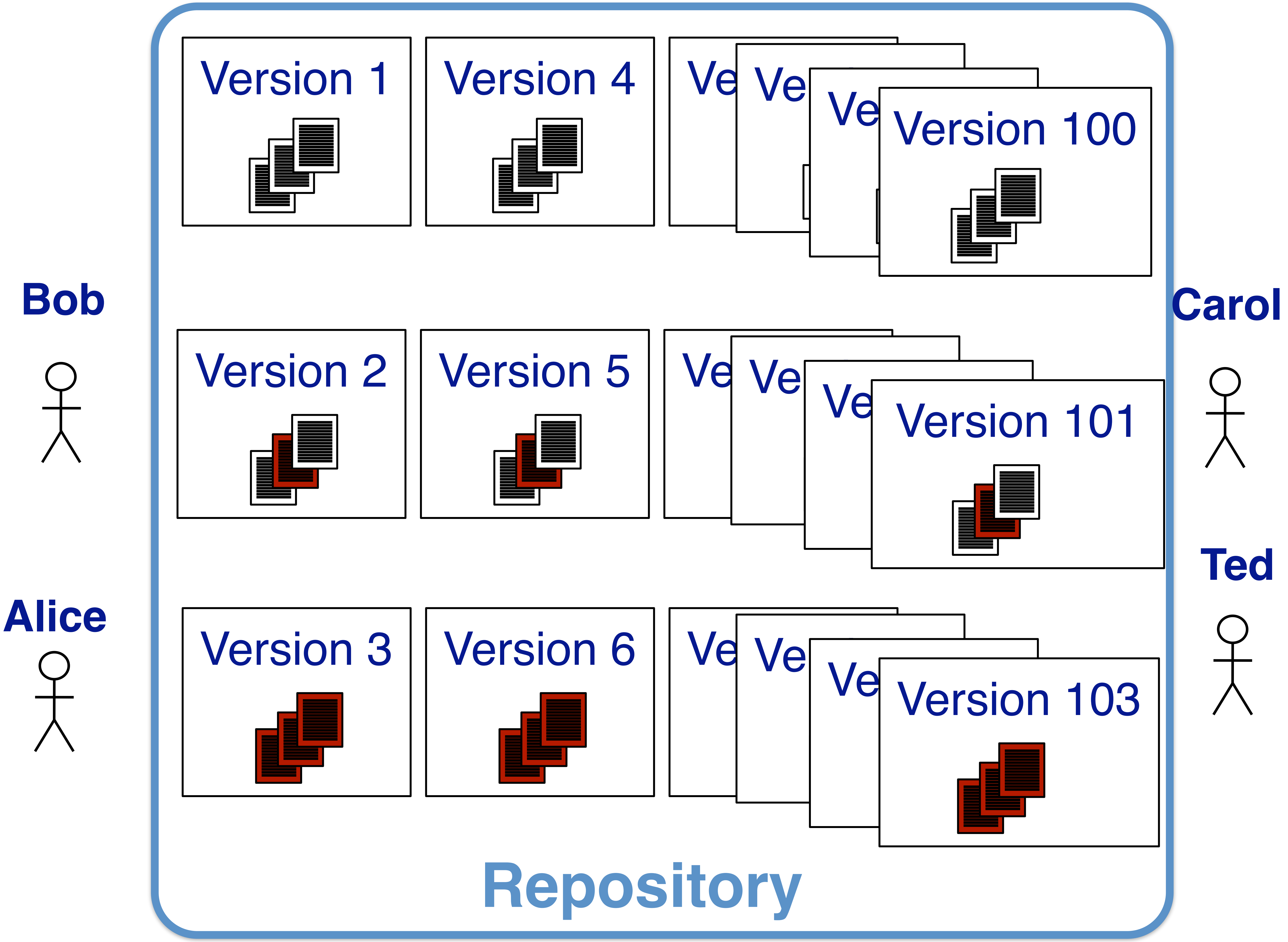
Version 2

Version 3

Snapshot

Carol

Repository



Bob

Alice

Carol

Ted

Version 1

Version 4

Ve

Ve

Ve

Version 100

Version 2

Version 5

Ve

Ve

Ve

Version 101

Version 3

Version 6

Ve

Ve

Ve

Version 103

Repository

git

git

Tracks the history of a collection of files

git

Tracks the history of a collection of files

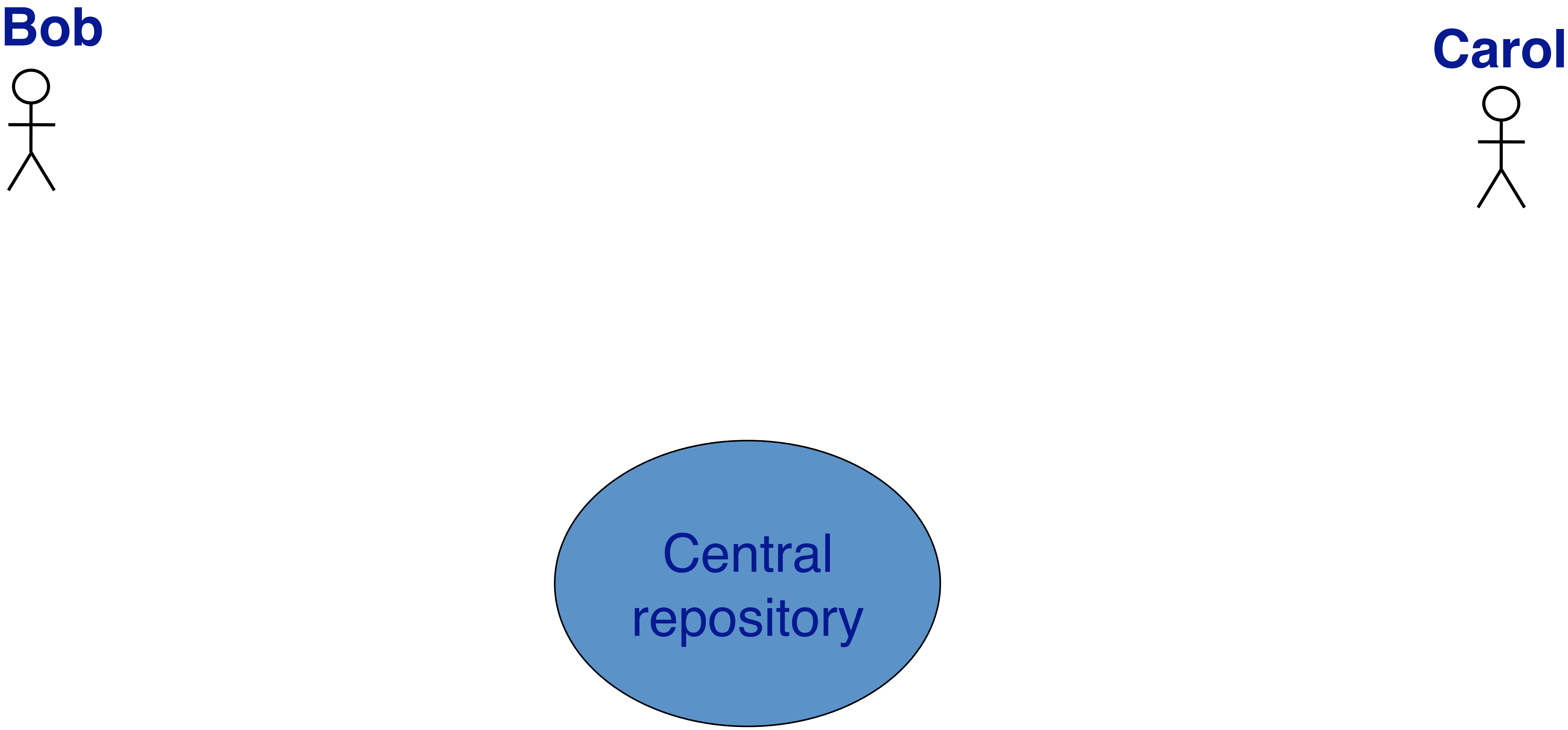
Can revert the collection of files to another version

git

distributed
version control
system

What is a distributed version control system?

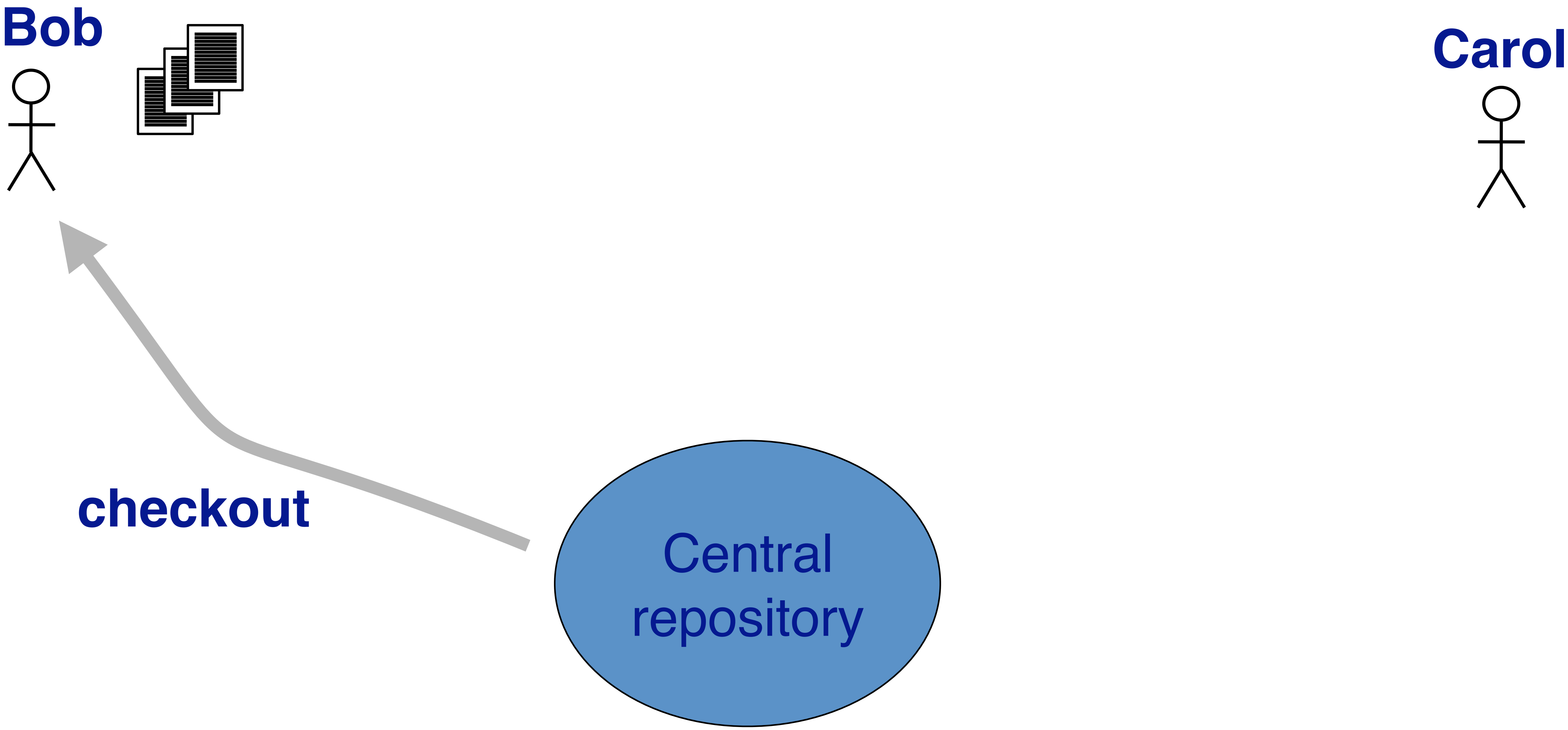
What is a centralized version control system?

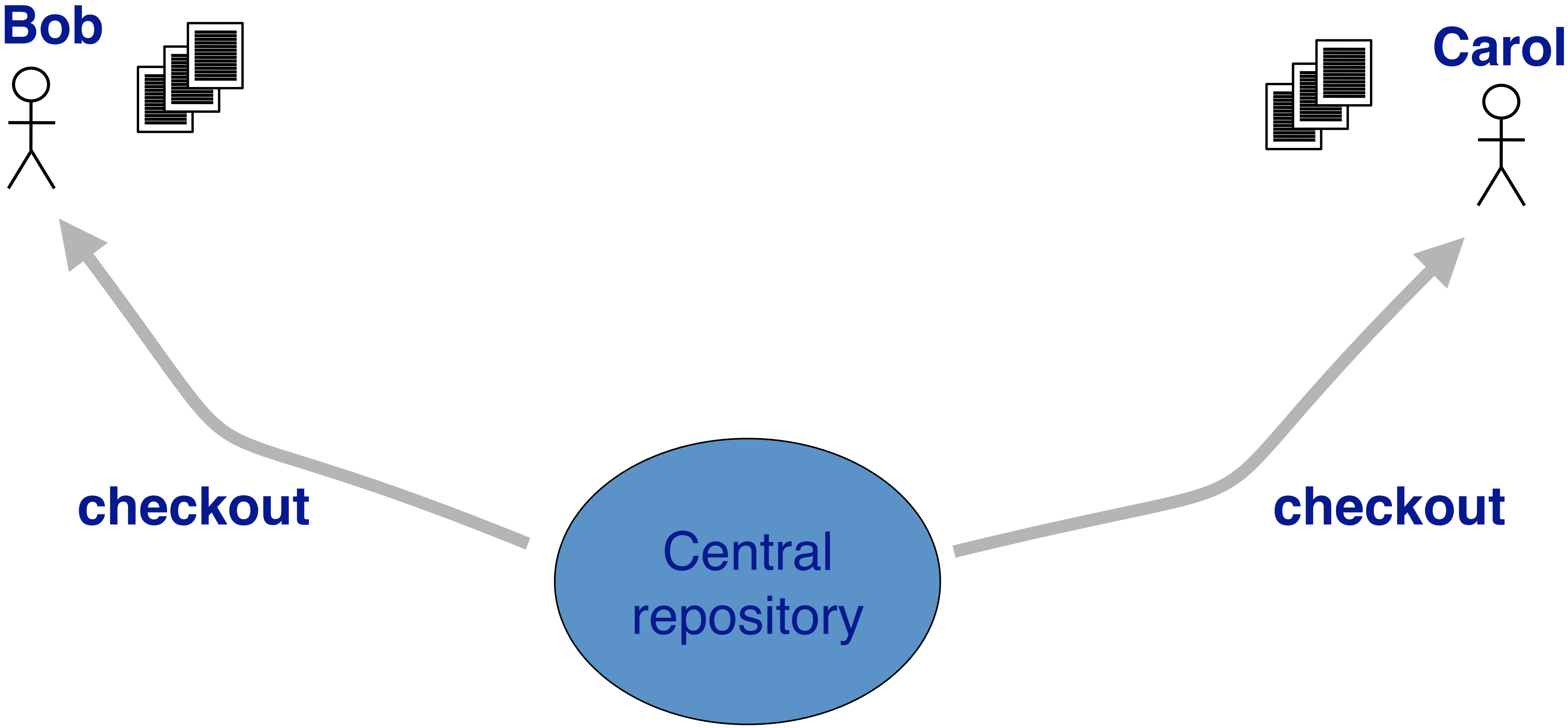


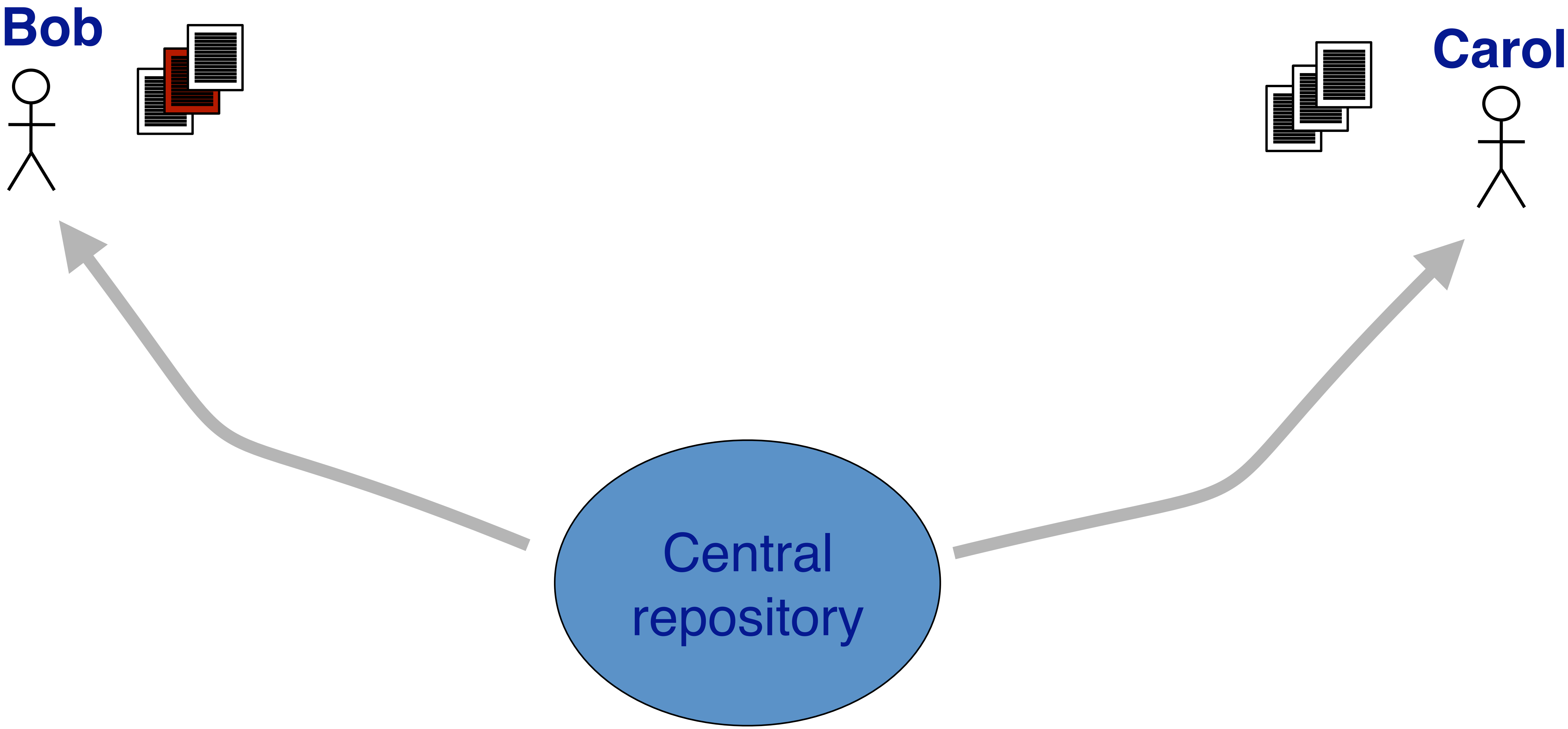
Bob

Carol

Central repository


Bob
Carol
checkout
Central repository

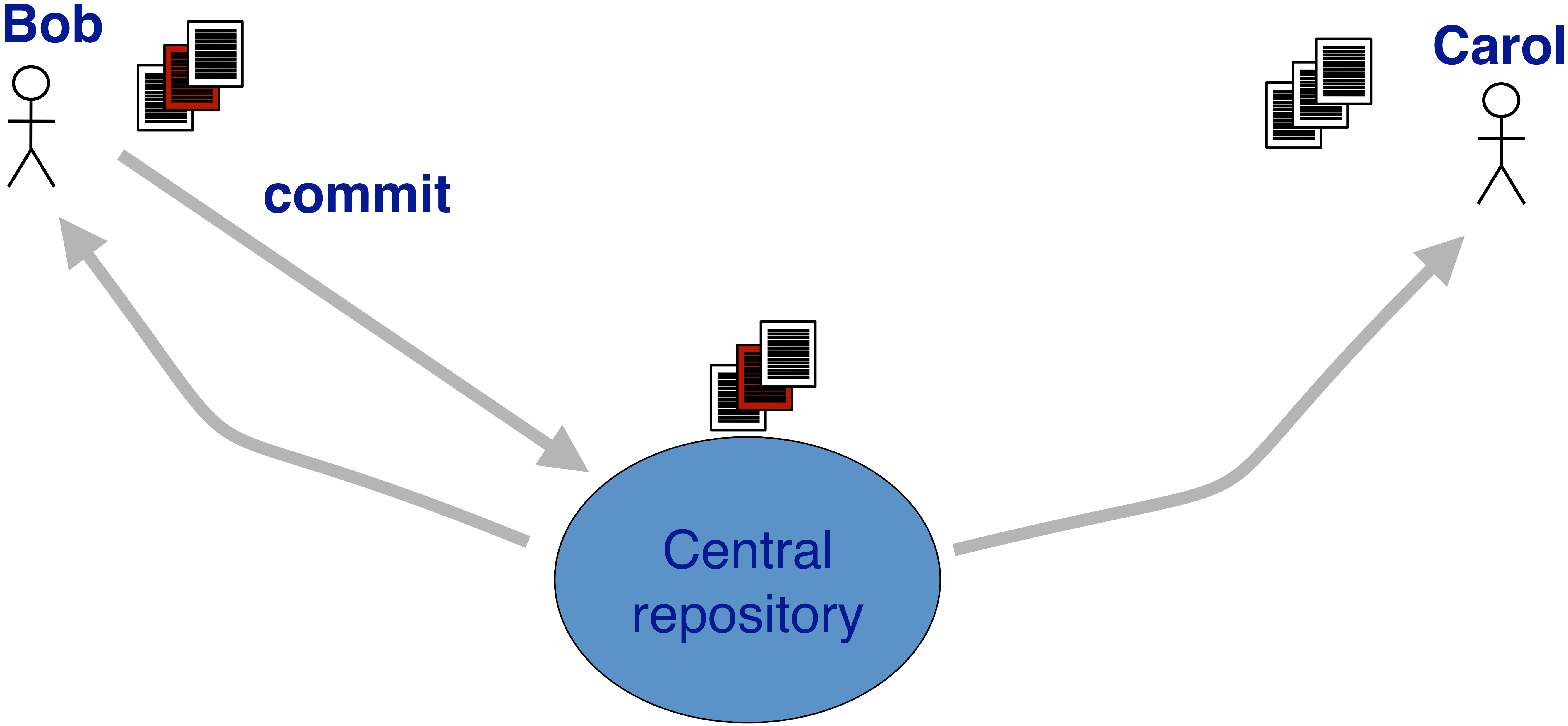

Bob
Carol
checkout
checkout
Central repository

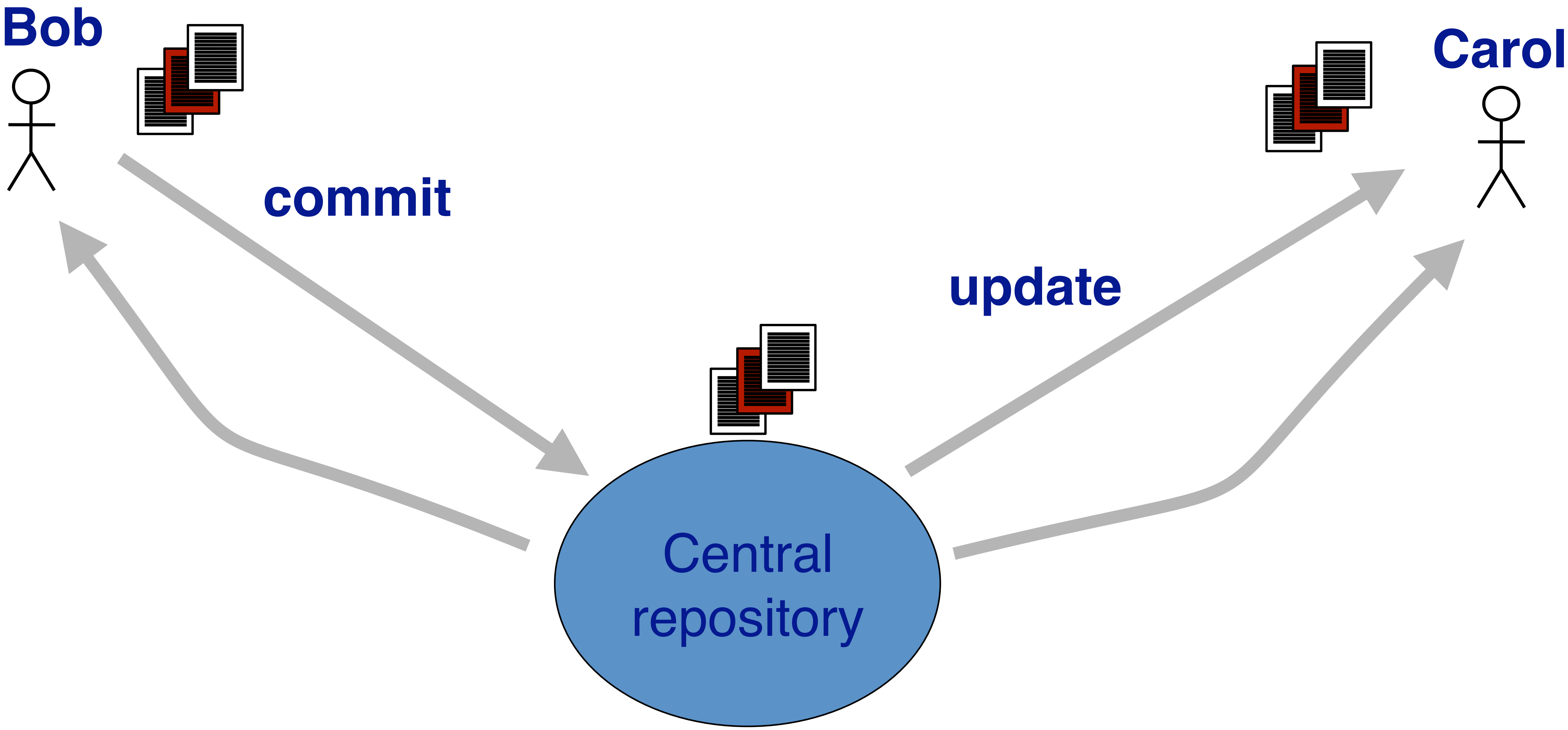


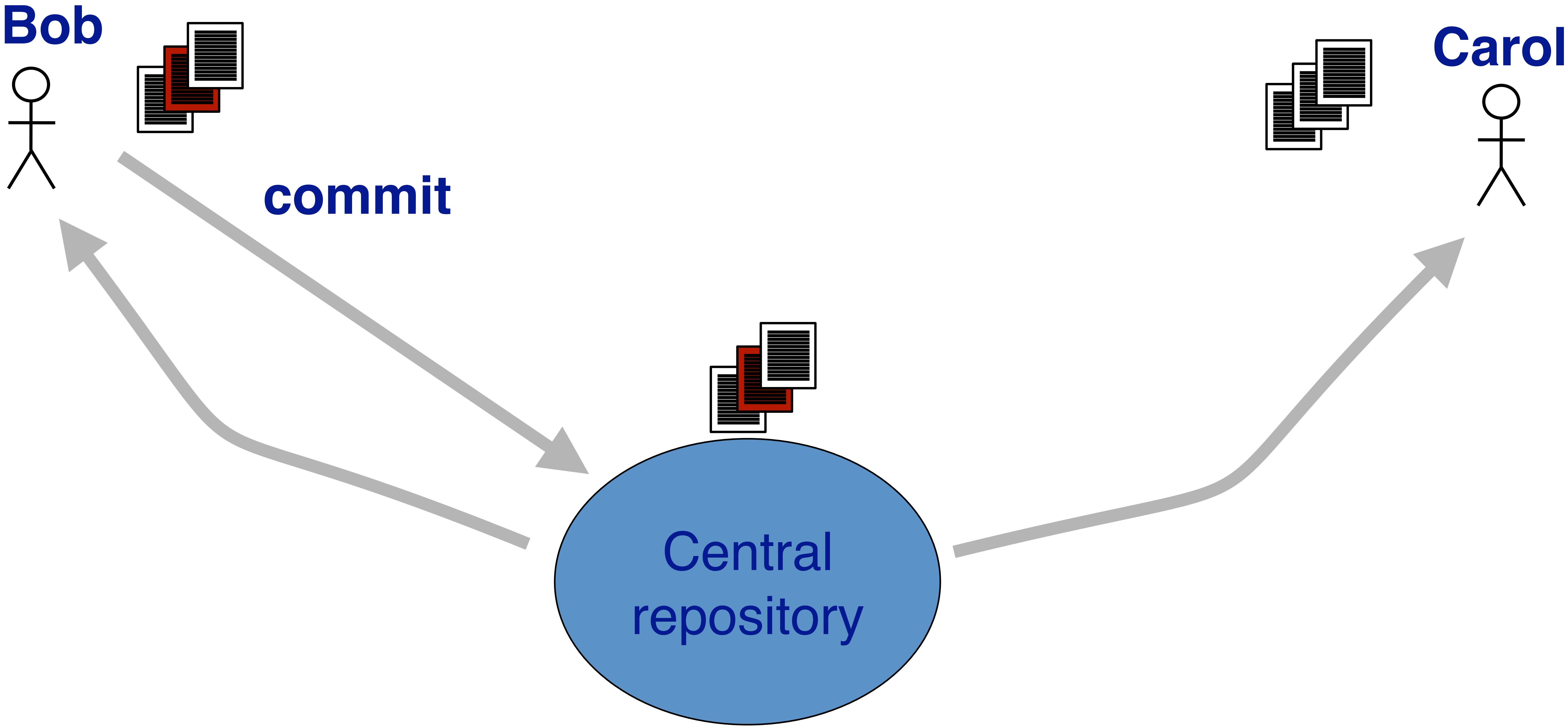
Bob

Carol

Central repository


Bob
commit
Carol
Central repository


Bob
commit
Carol
update
Central repository

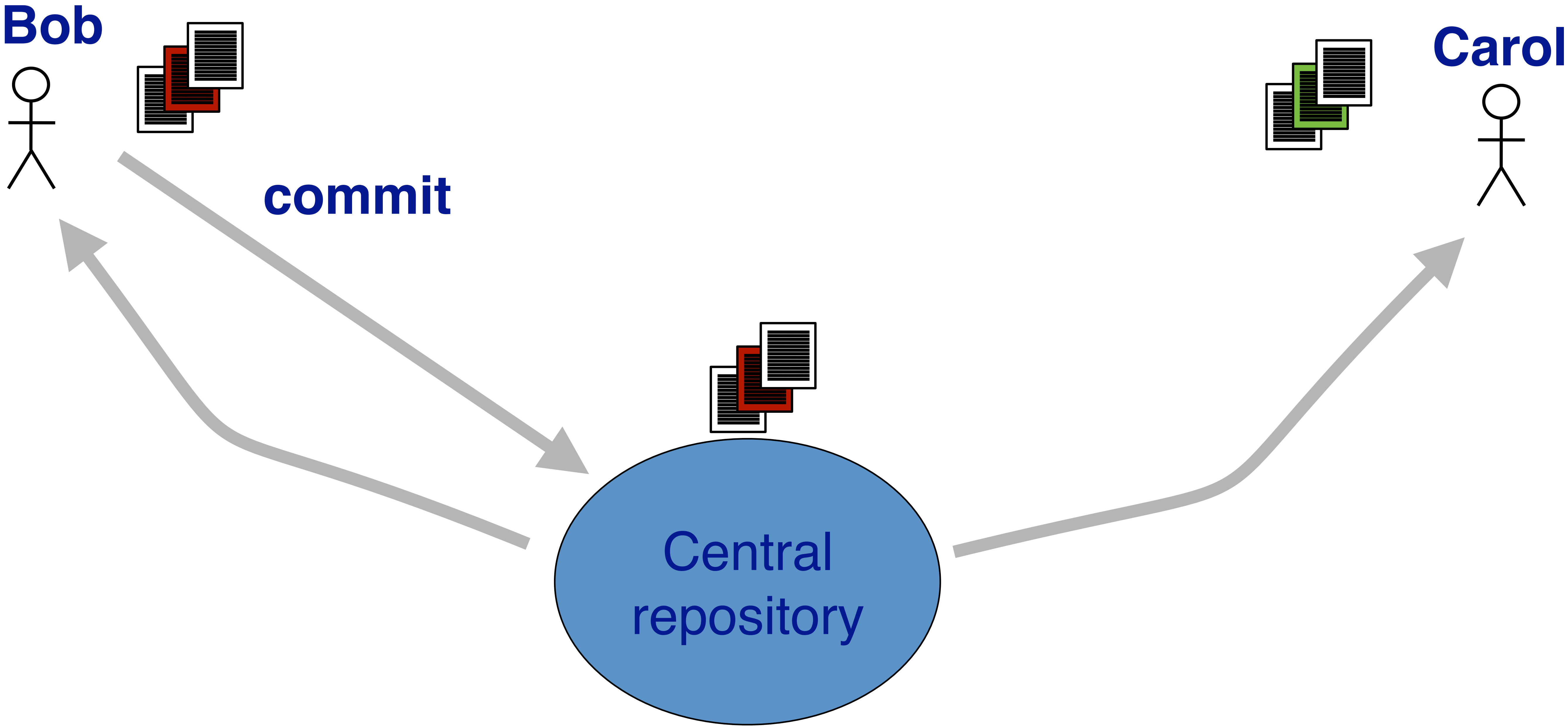


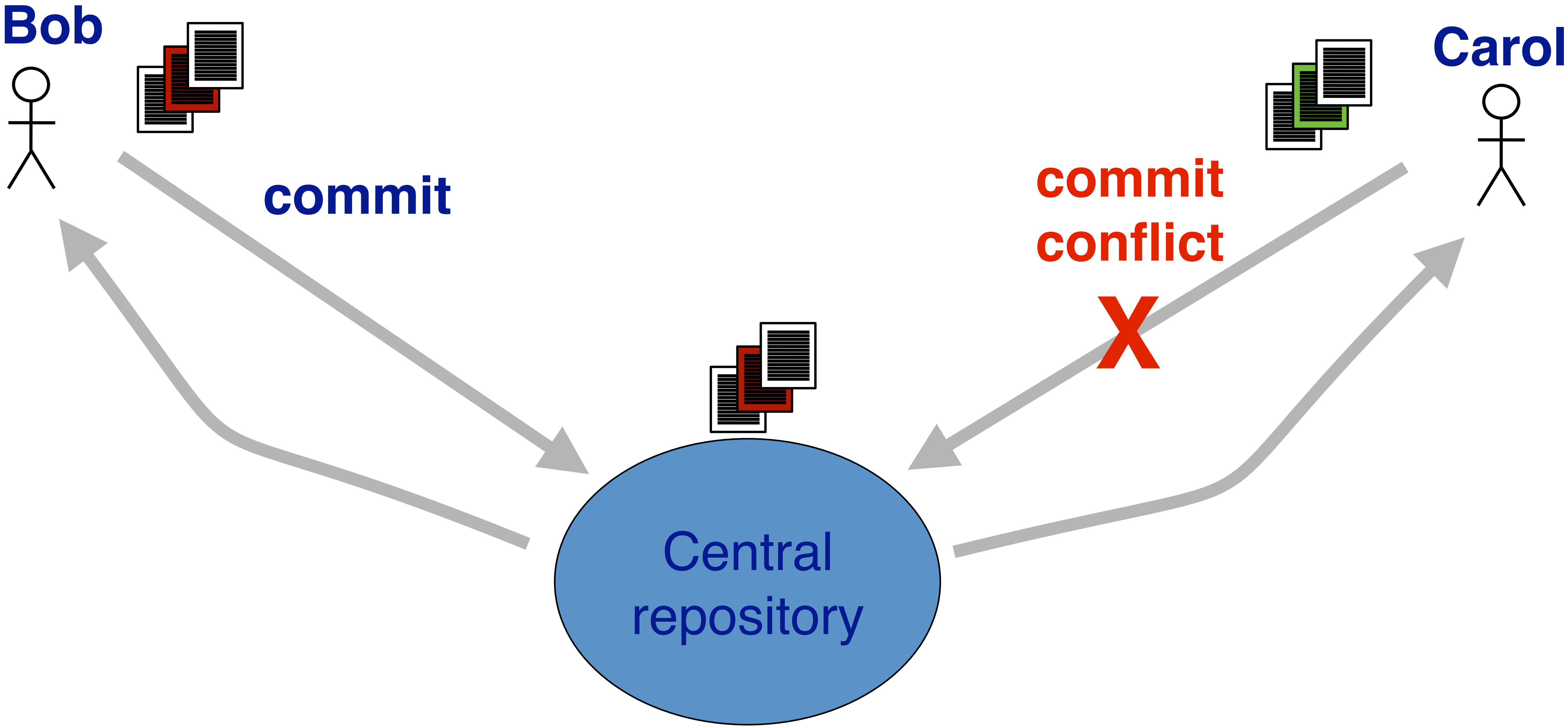
Bob

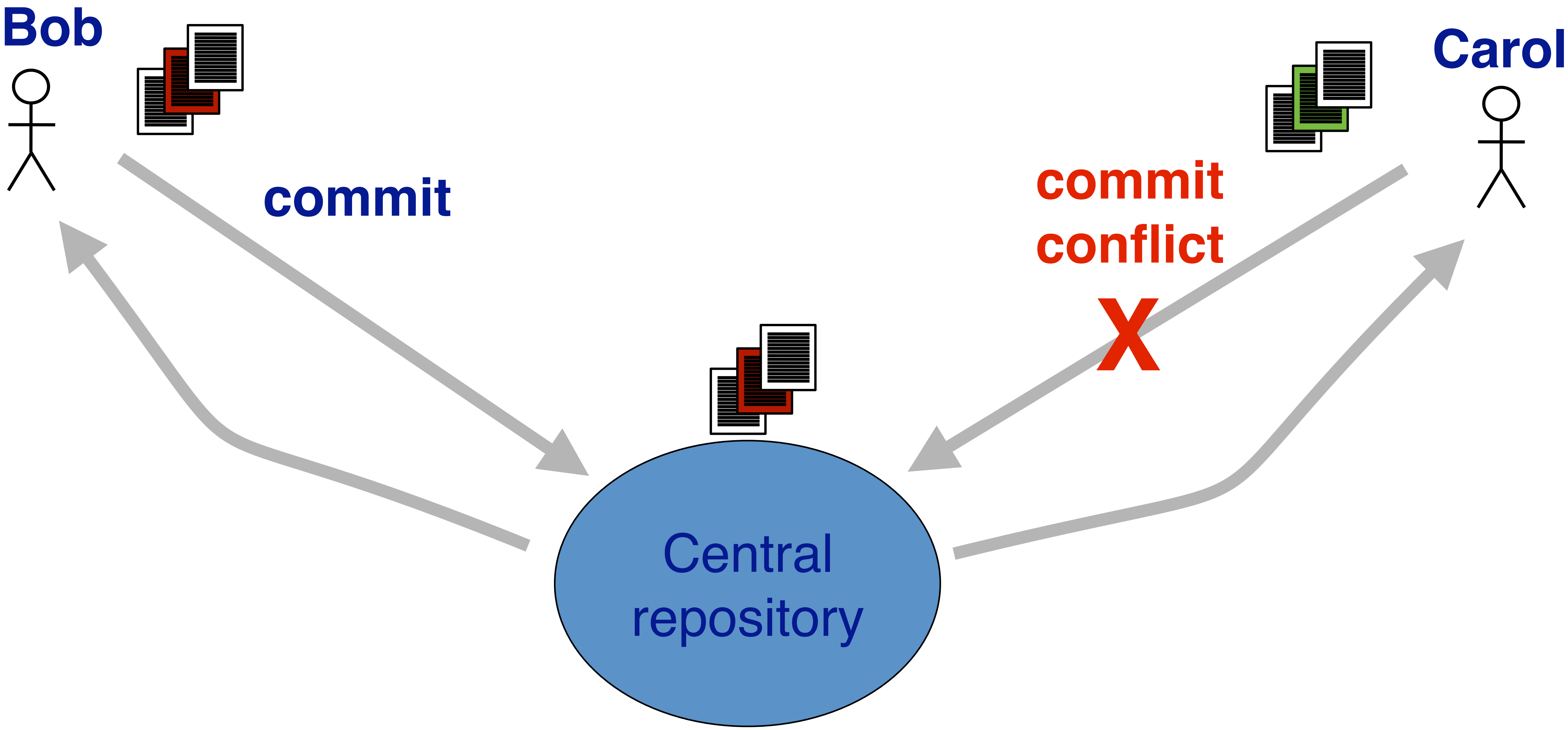
commit

Carol

Central repository


Bob
commit
Carol
Central repository


Bob
commit
commit
conflict
X
Carol
Central
repository



Bob

commit

Carol

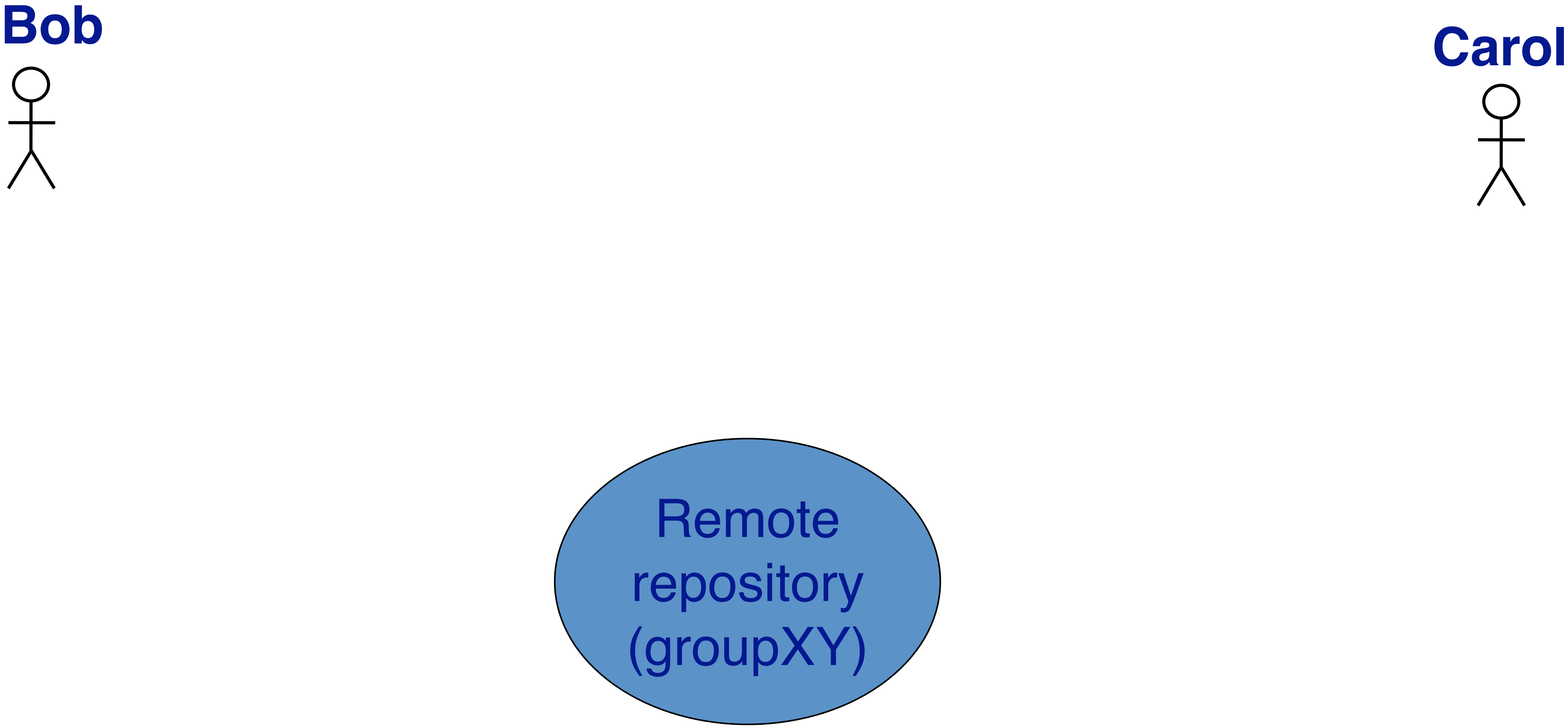
commit conflict

X

Central repository

you must update before every commit

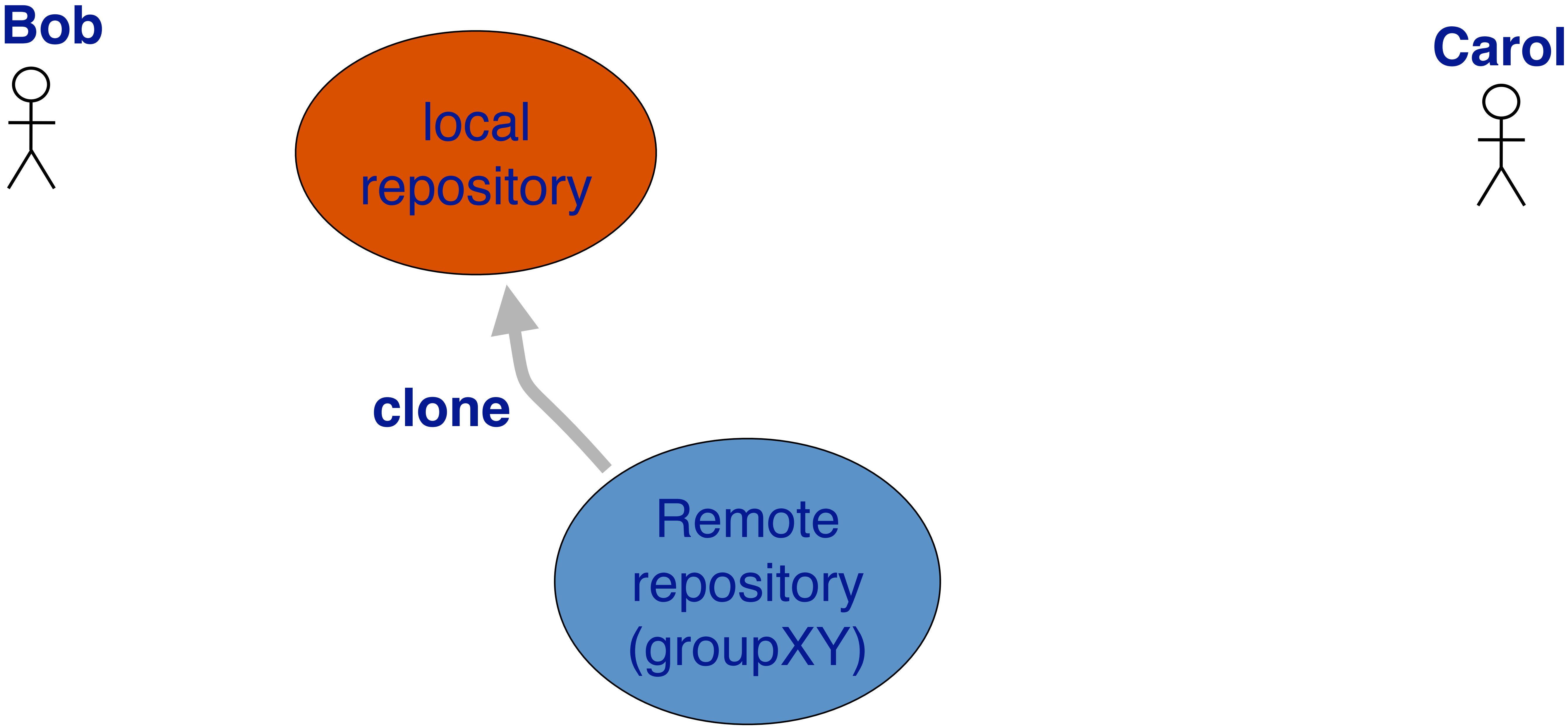
What is a distributed version control system?



Bob

Carol

Remote repository (groupXY)



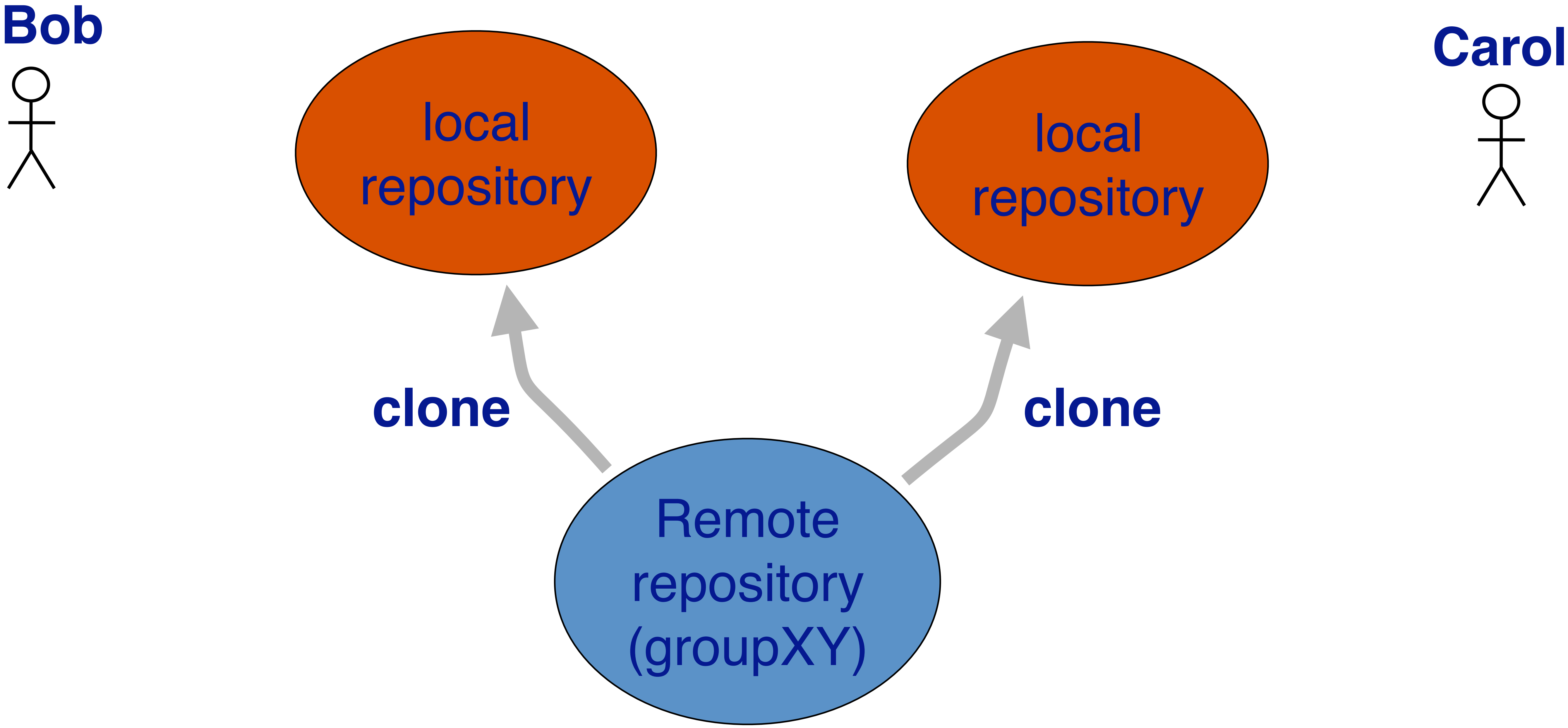
Bob

local repository

clone

Remote repository (groupXY)

Carol



Bob

Carol

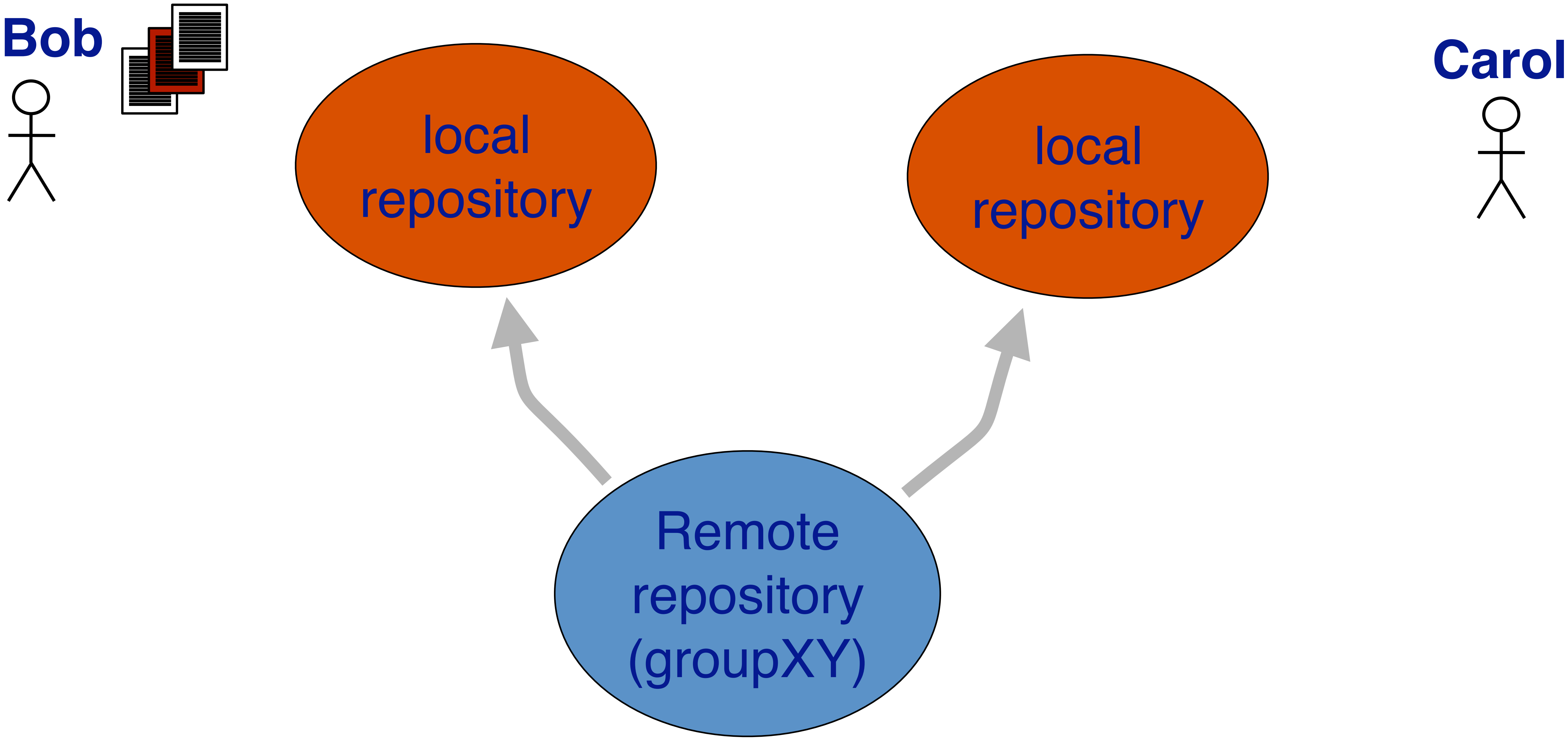
local repository

local repository

clone

clone

Remote repository (groupXY)



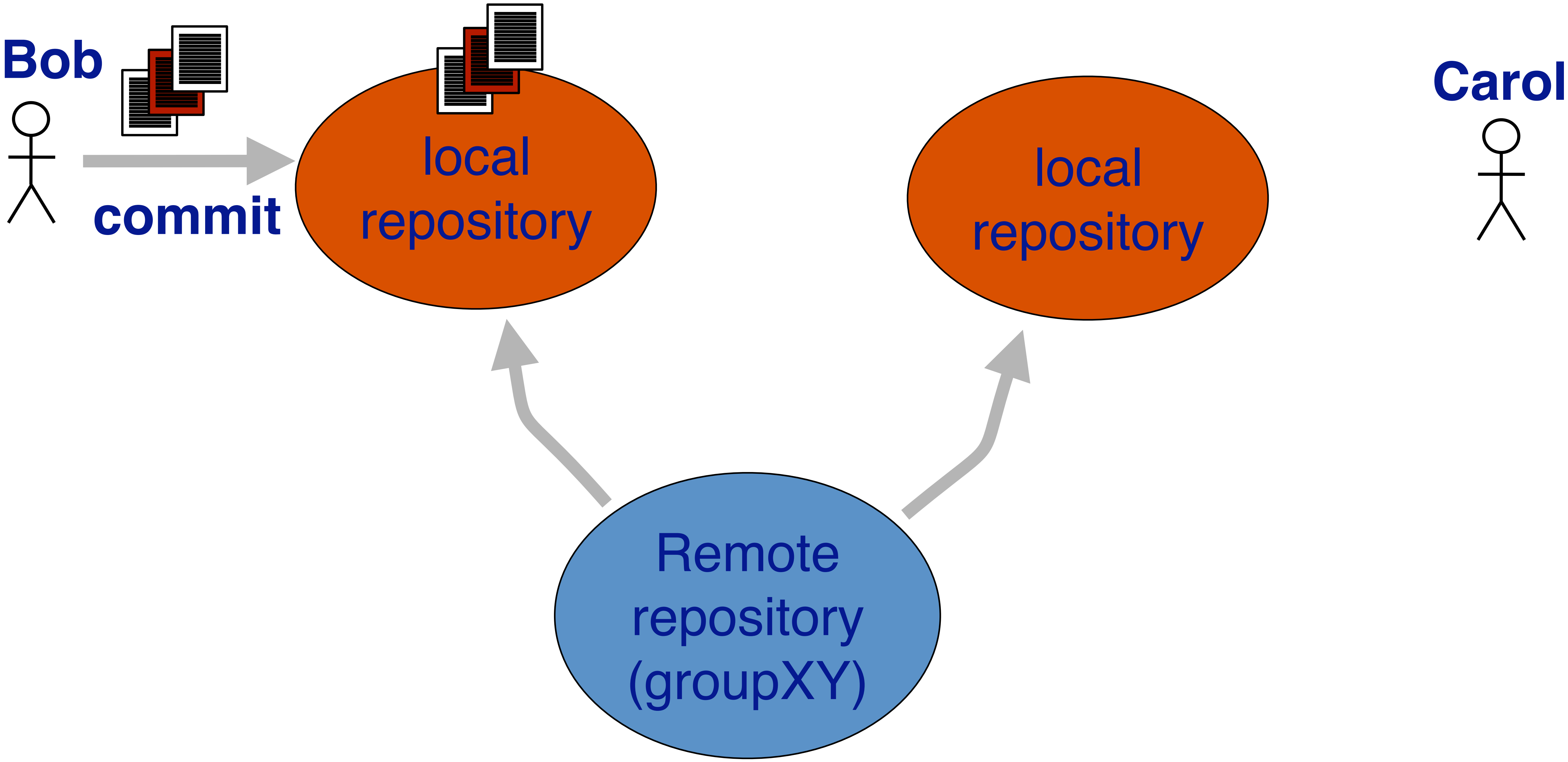
Bob

Carol

local repository

local repository

Remote repository (groupXY)



Bob

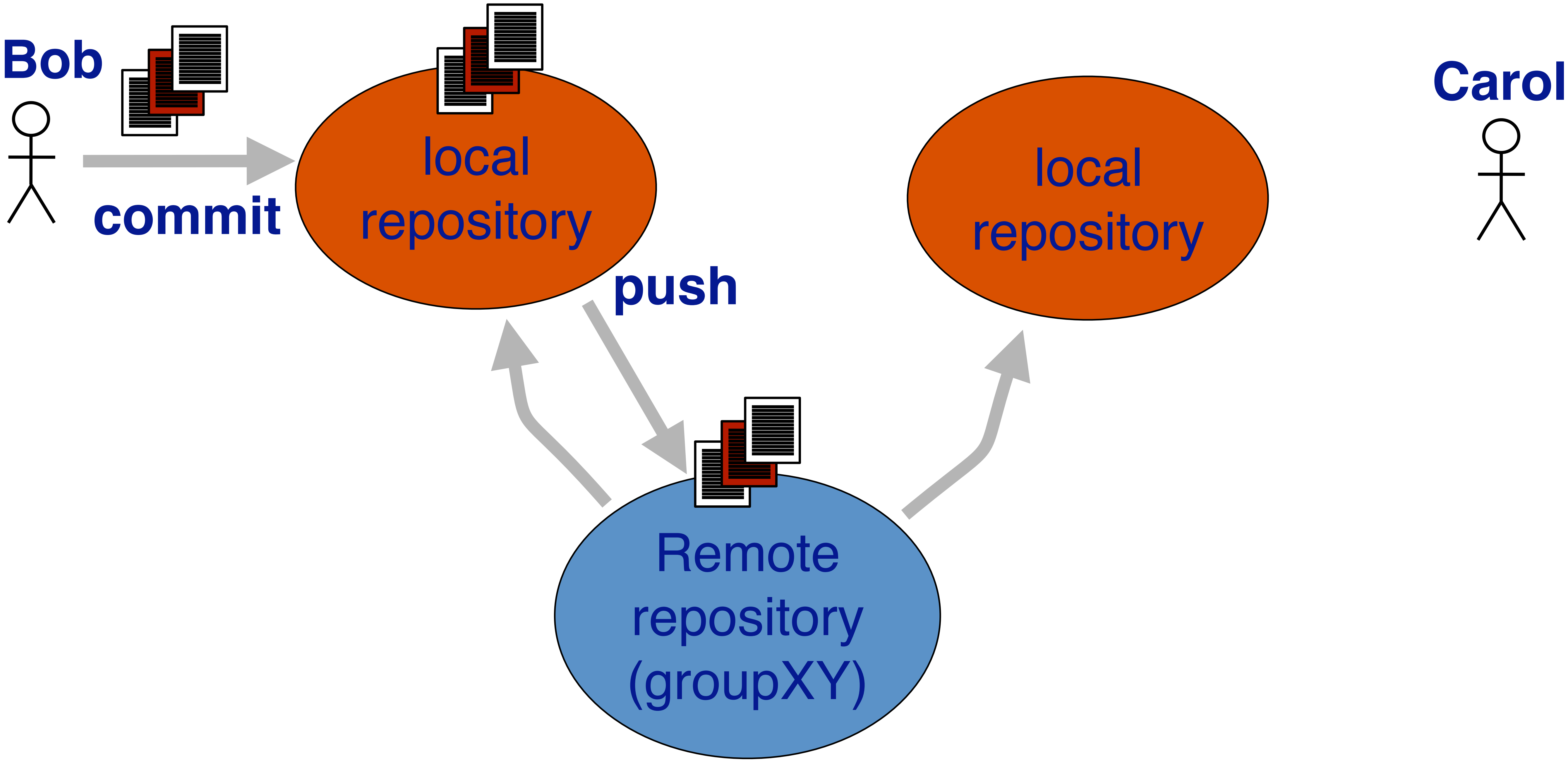
commit

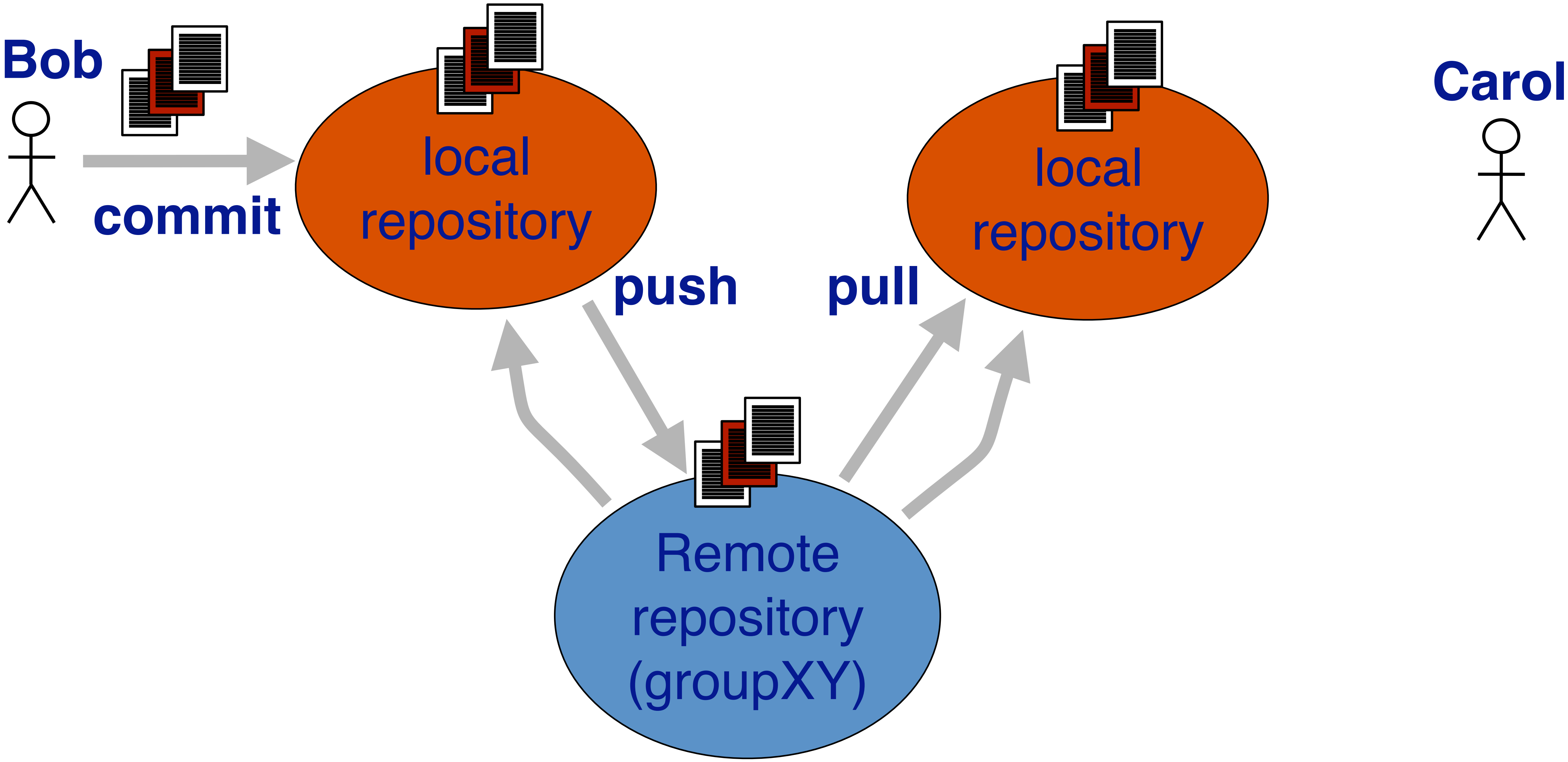
local repository

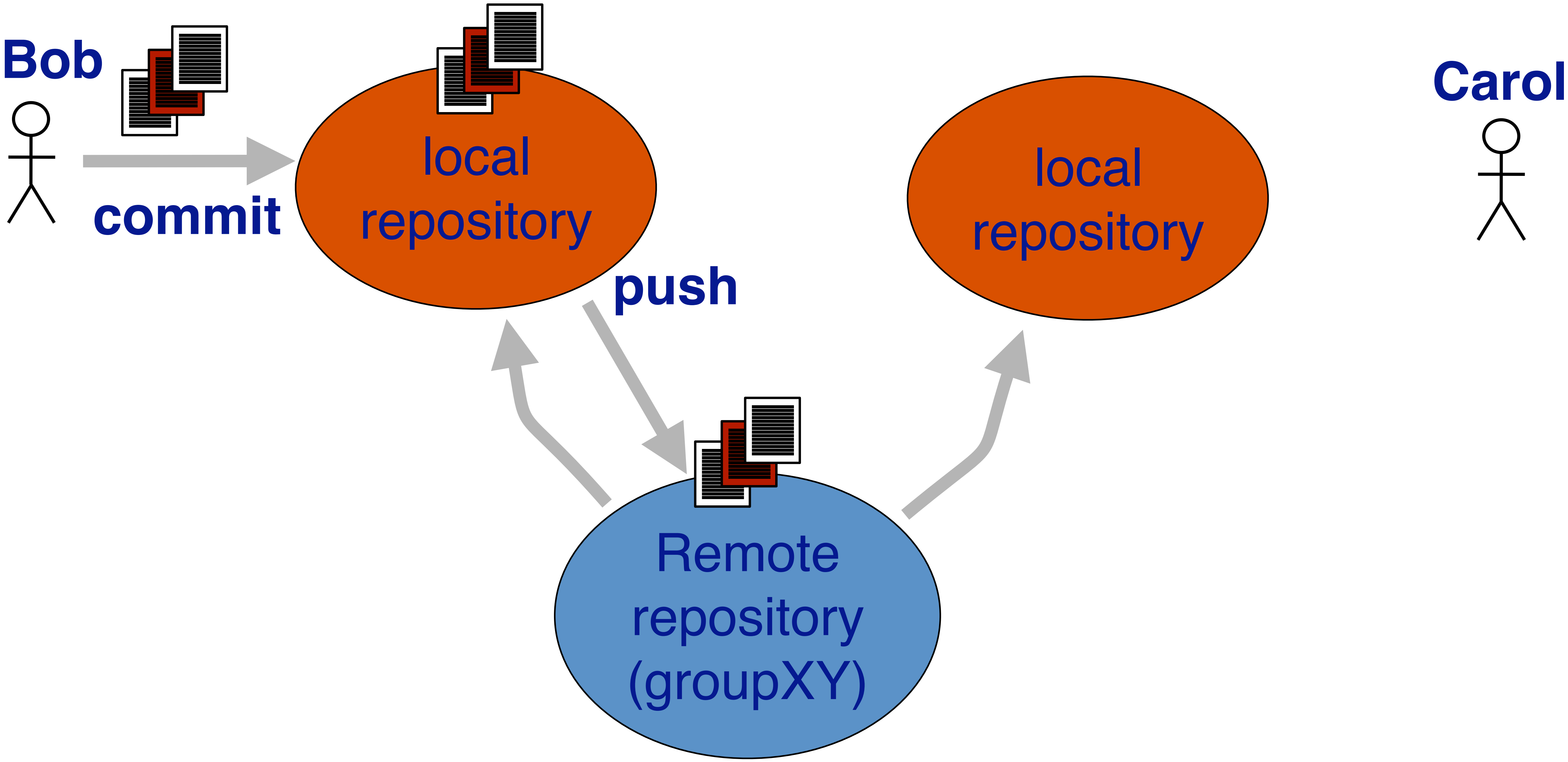
local repository

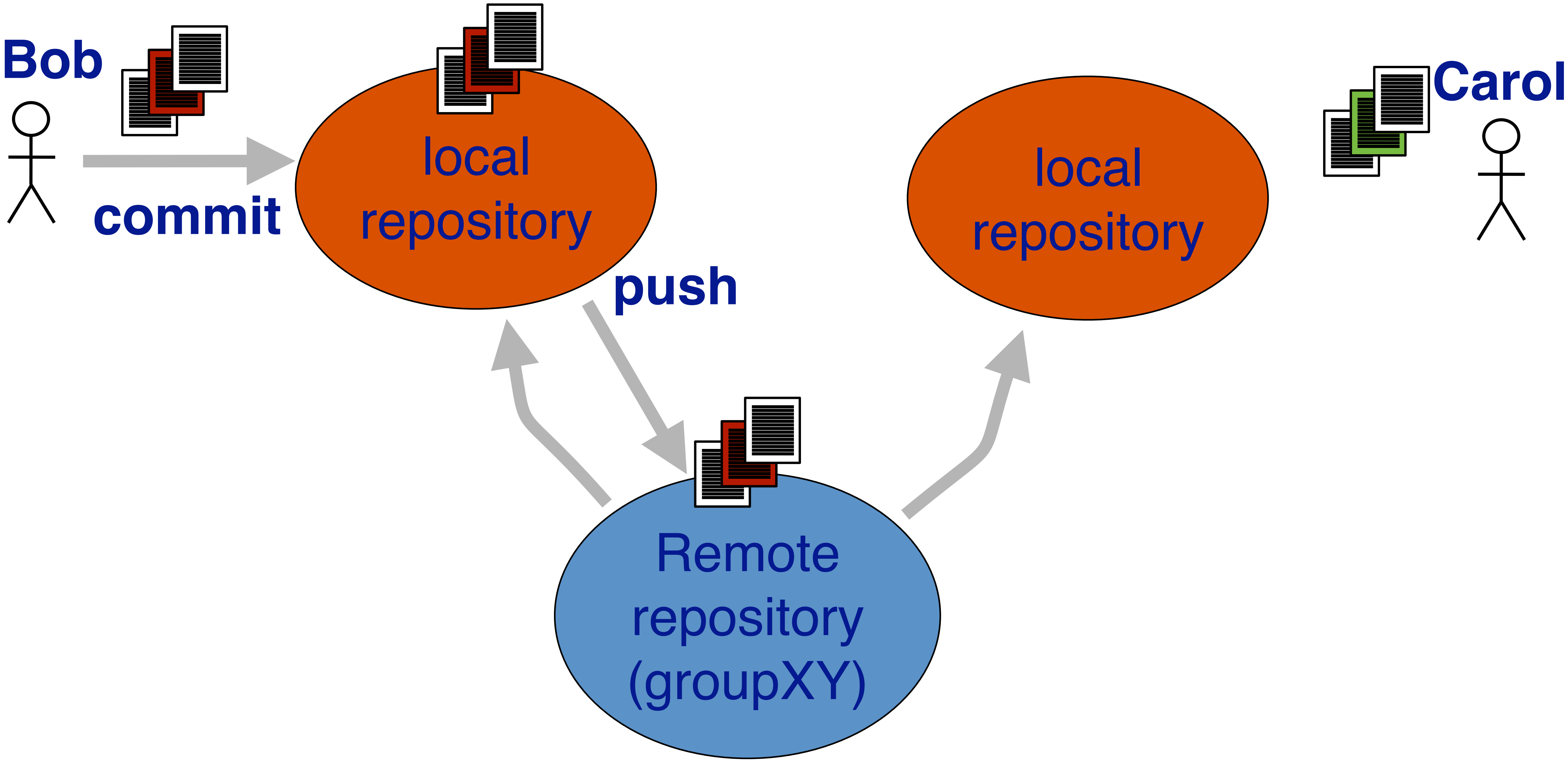
Carol

Remote repository (groupXY)


Bob
Carol
commit
local repository
local repository
push
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

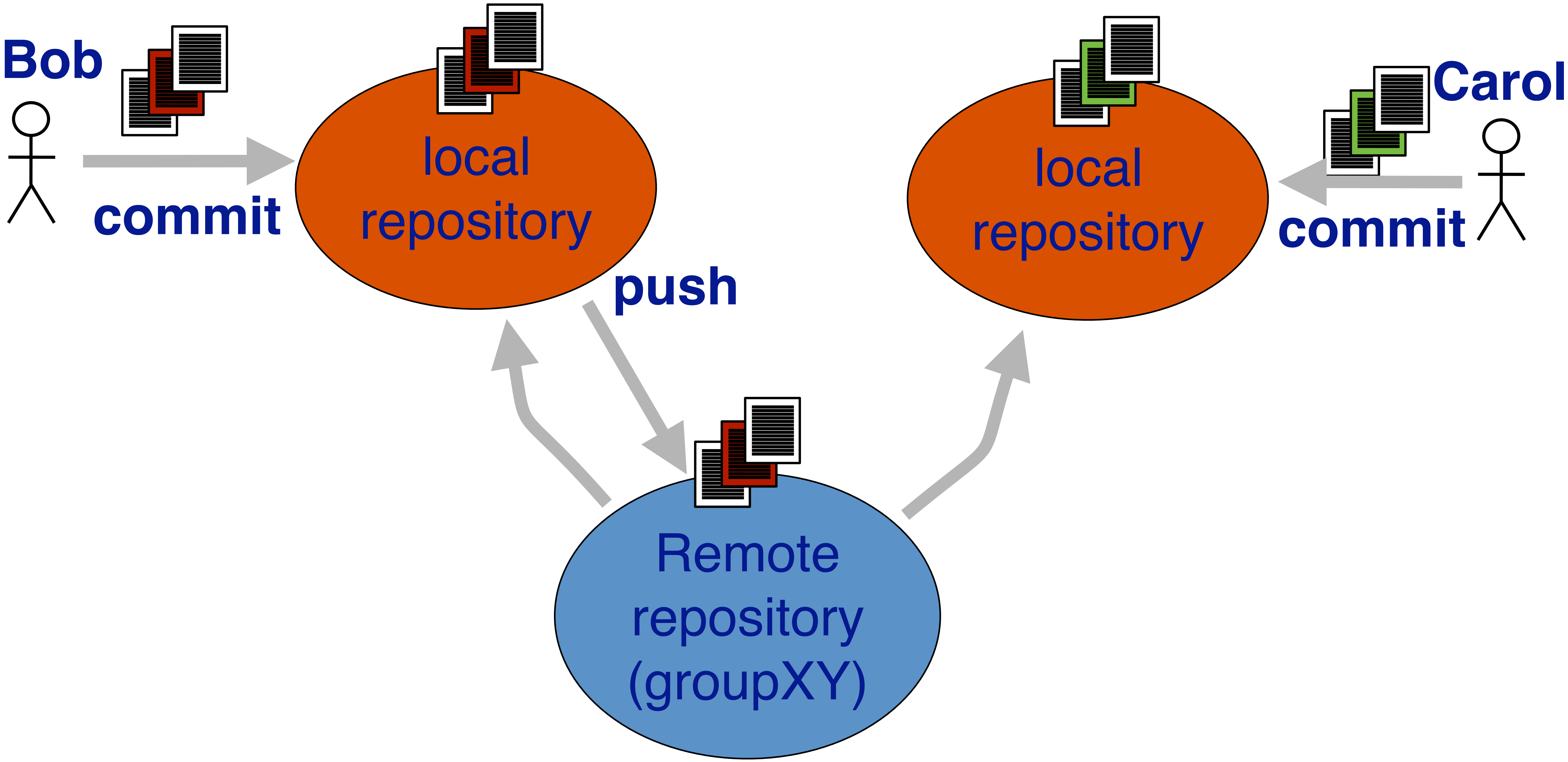
local repository

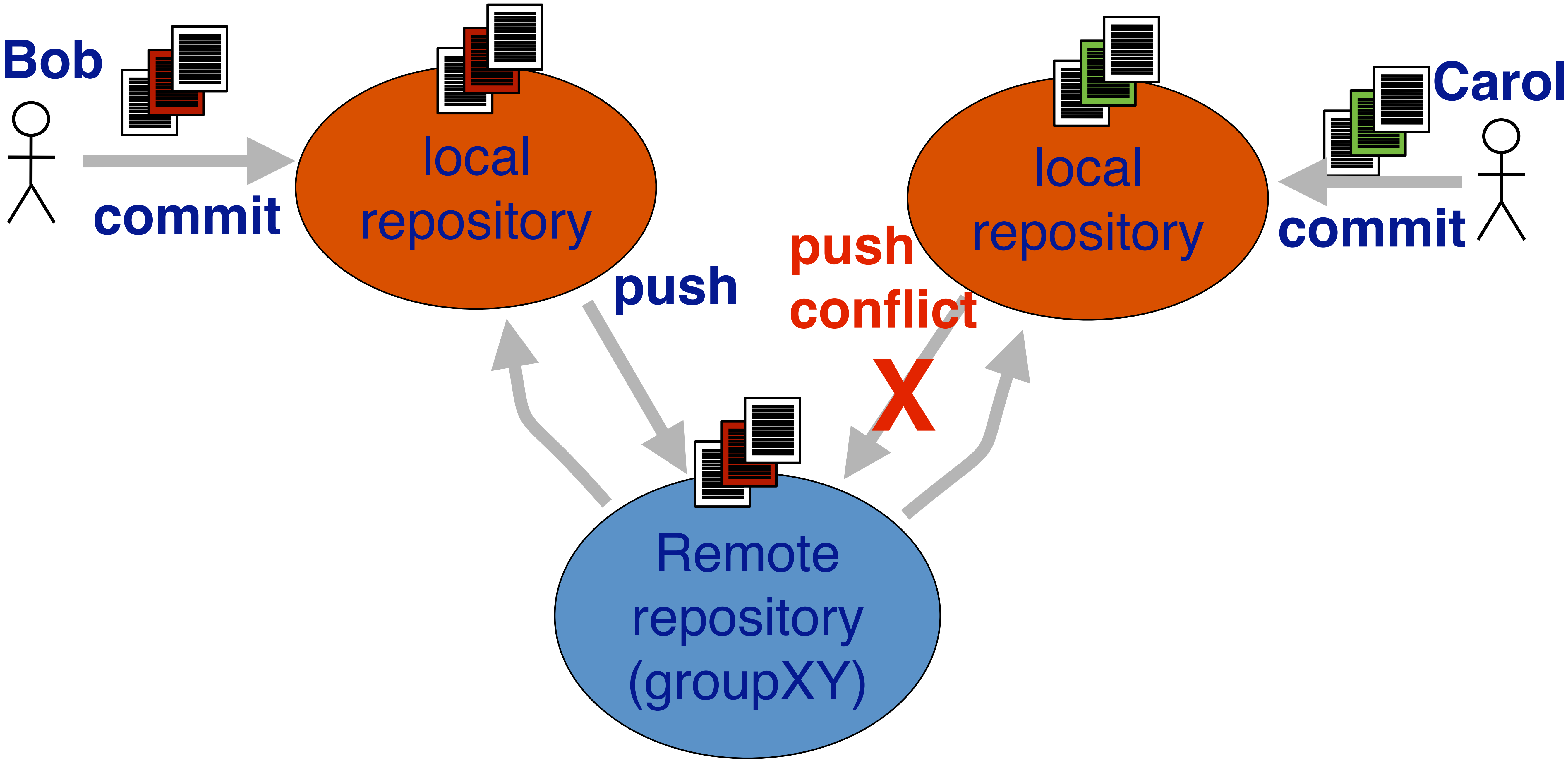
push

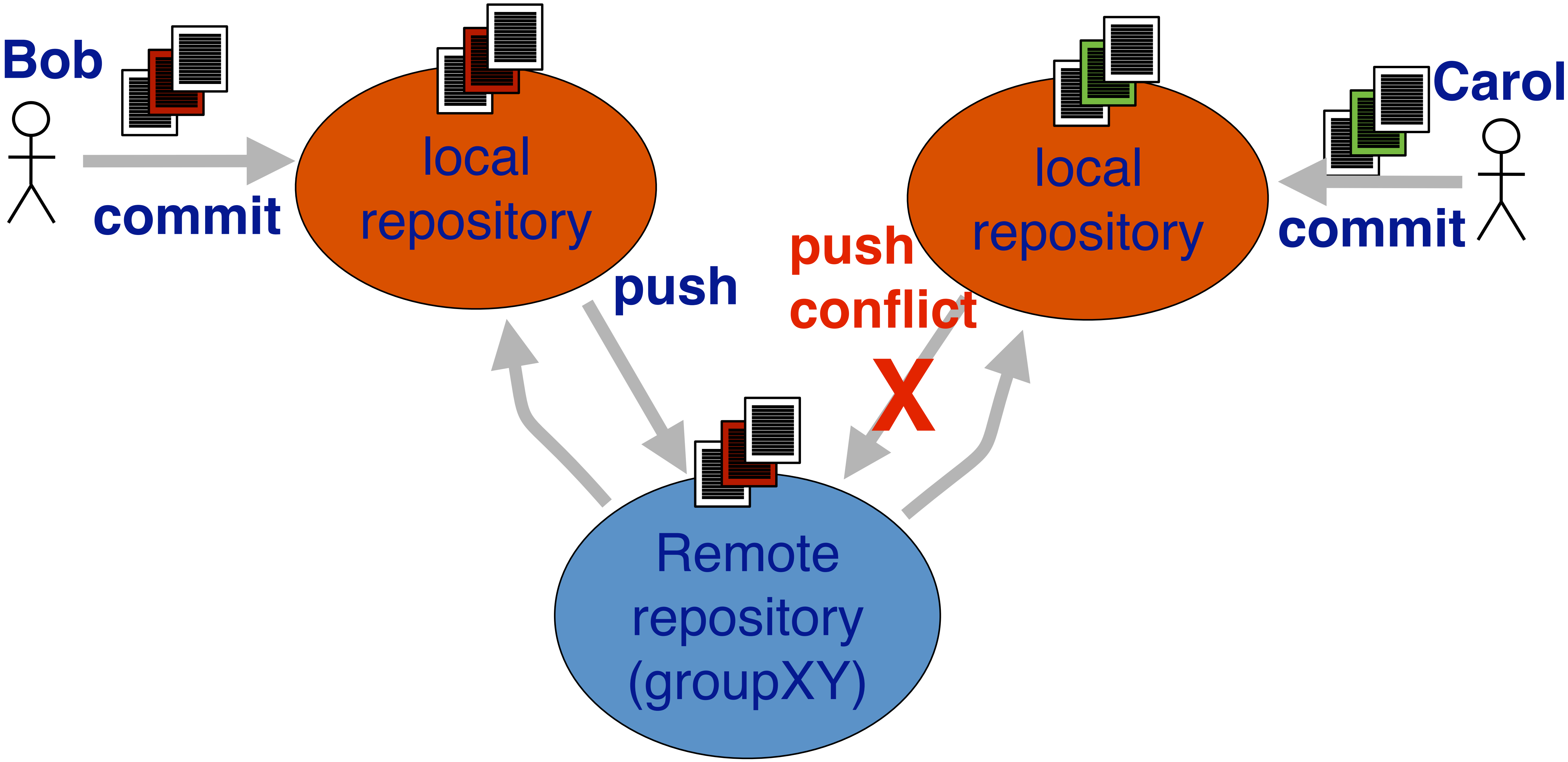
local repository

Carol

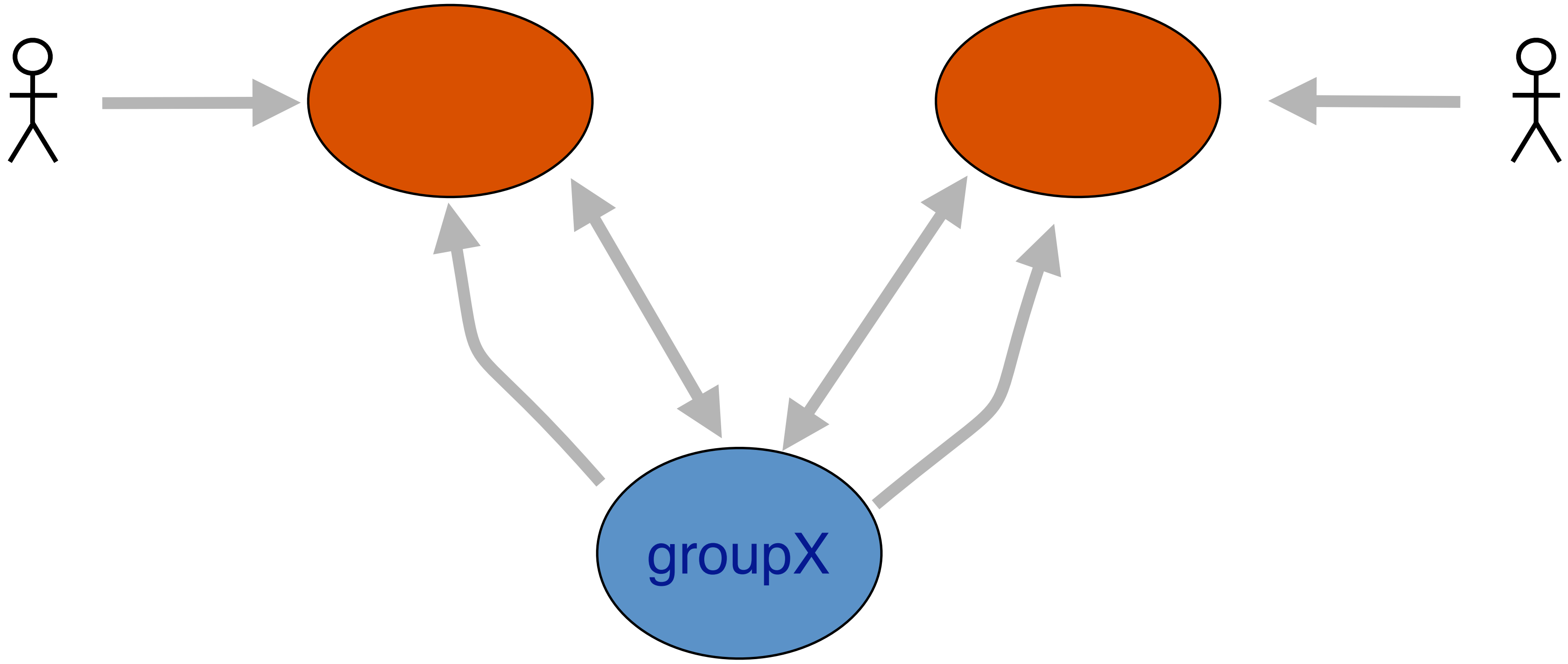
Remote repository (groupXY)

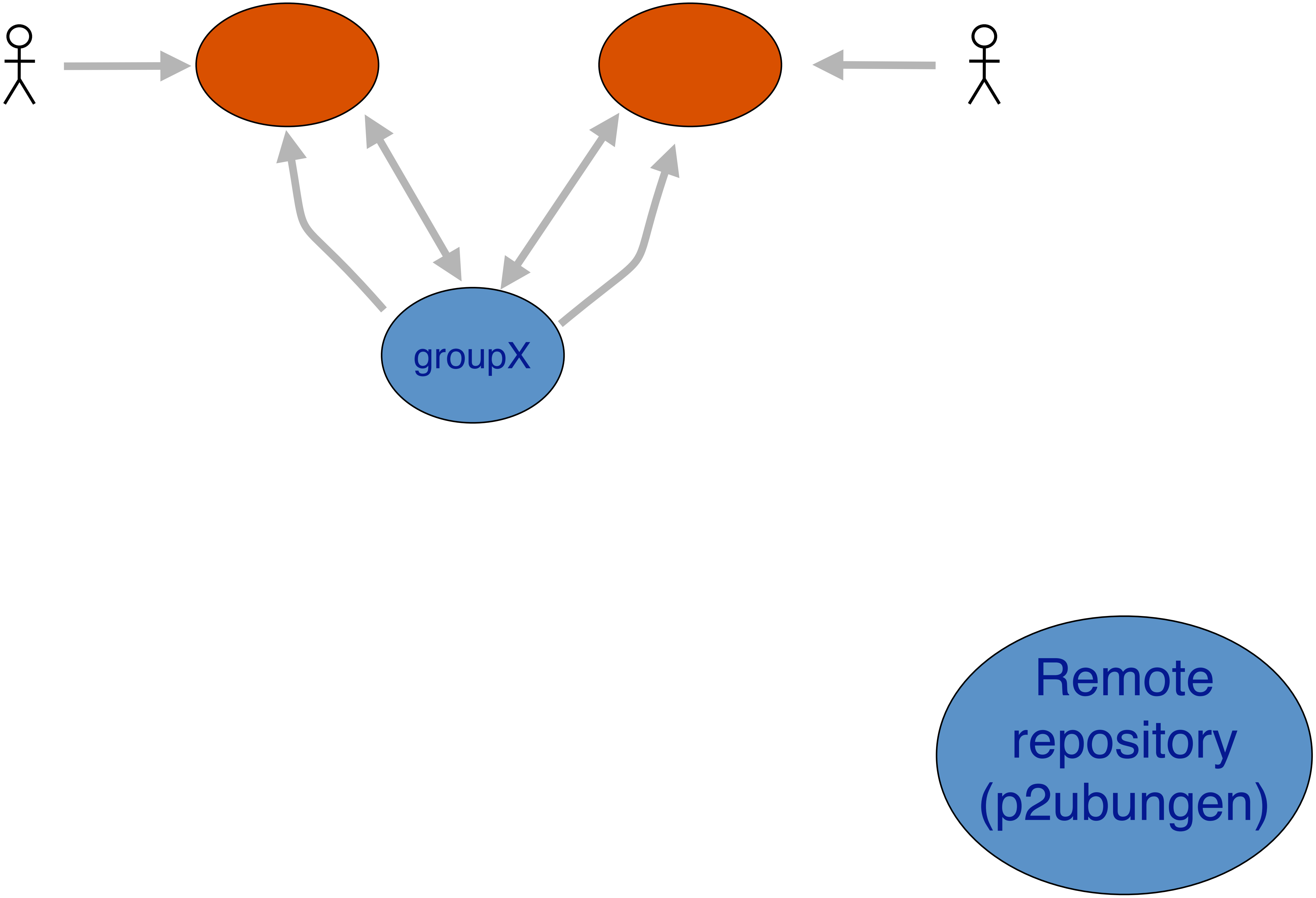

Bob
commit
local
repository
push
Carol
commit
local
repository
Remote
repository
(groupXY)


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



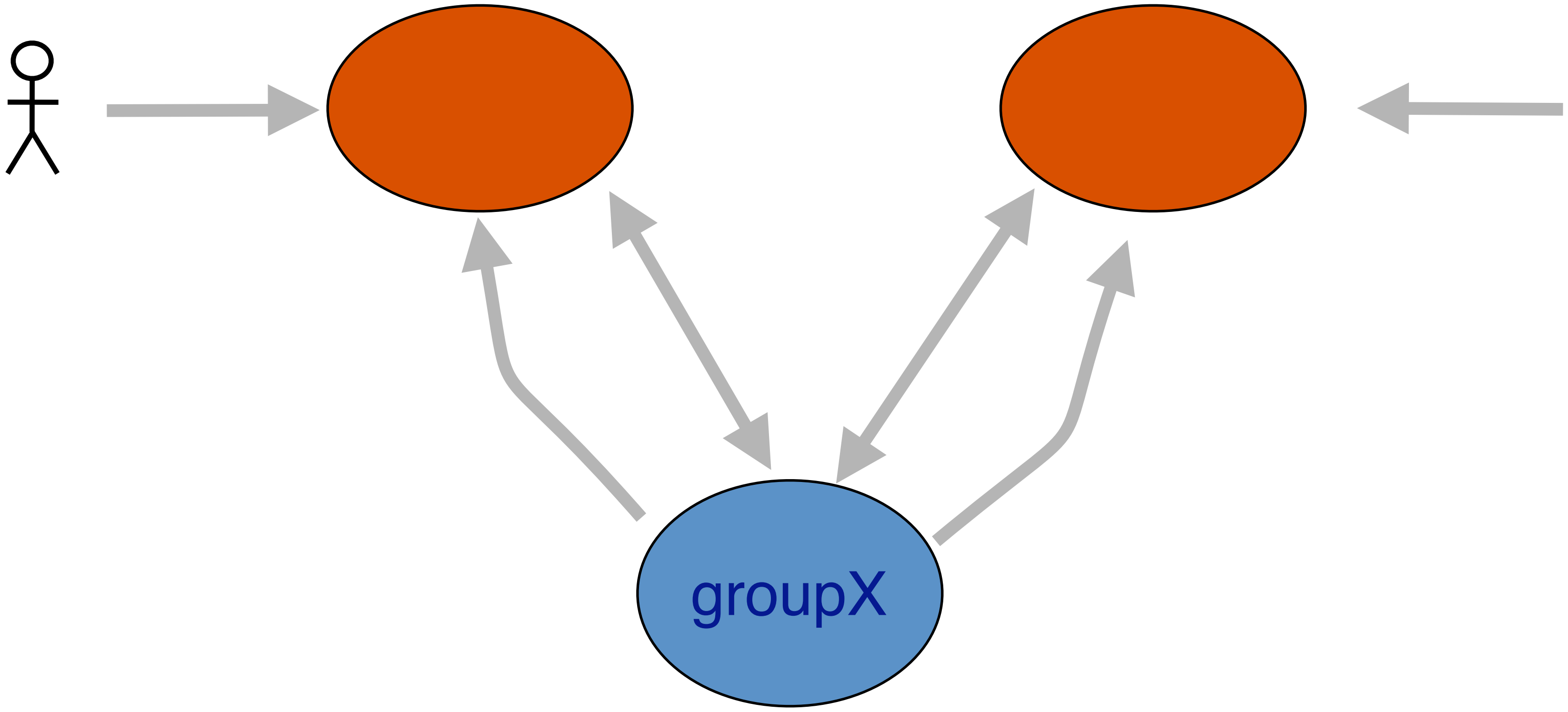
you must pull before every push


groupX

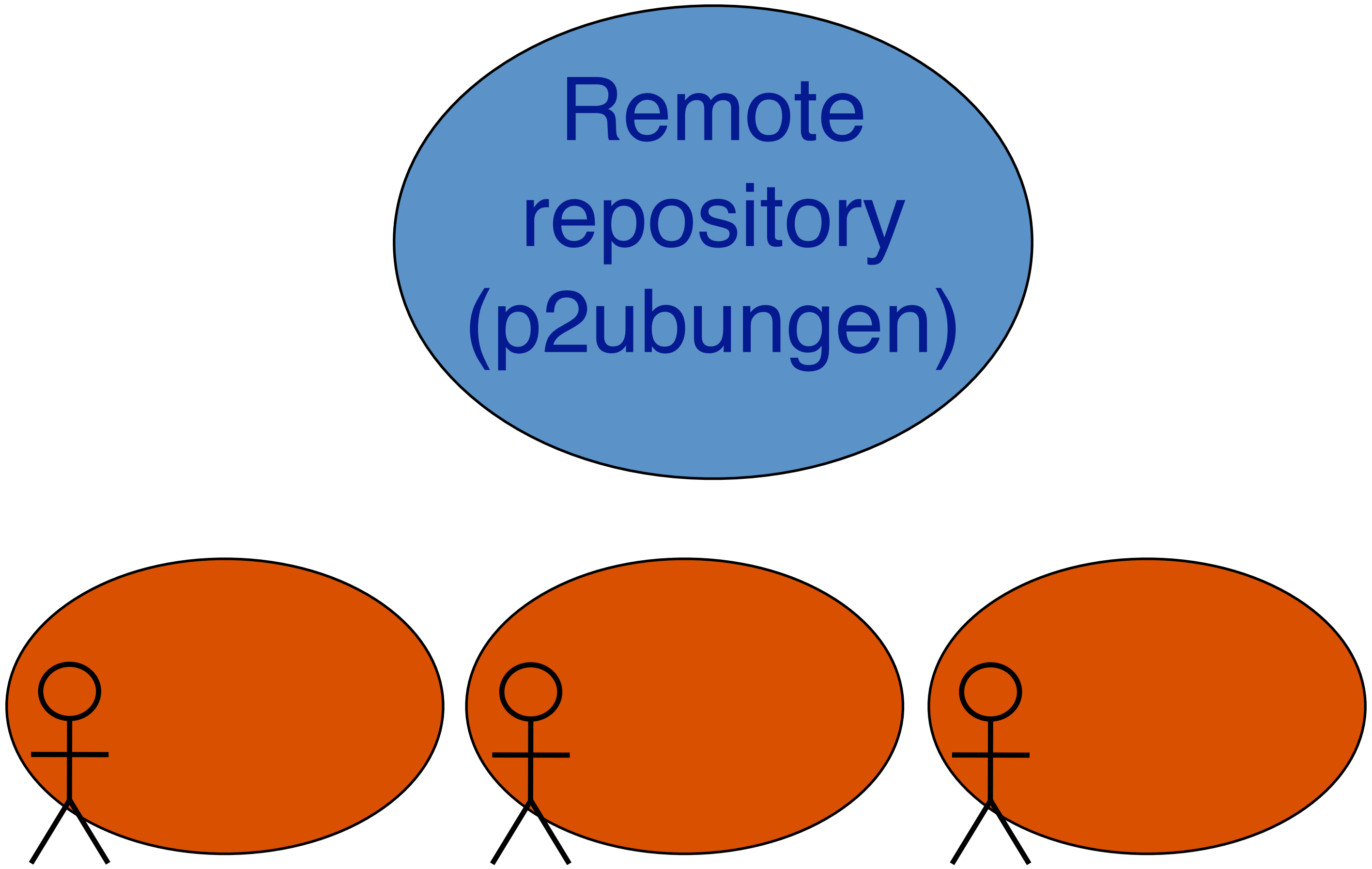


groupX

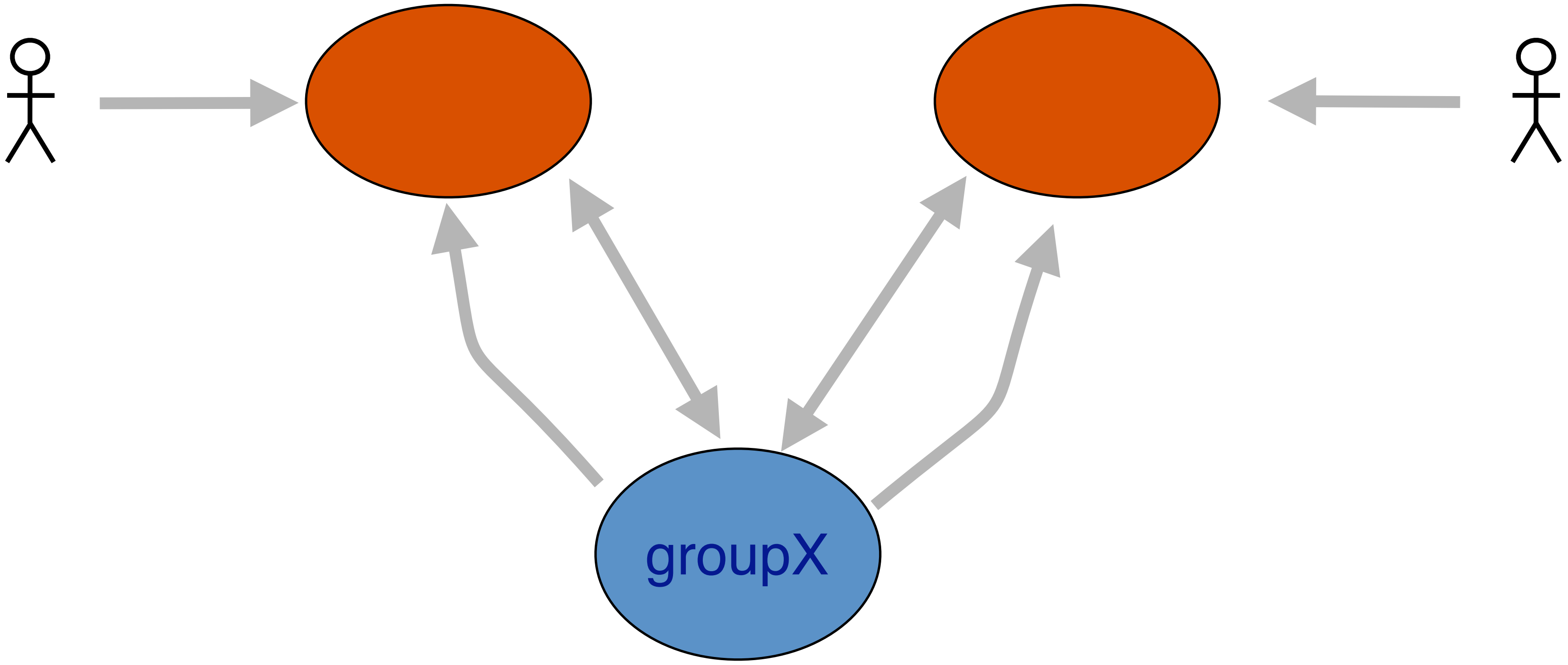
Remote repository (p2ubungen)

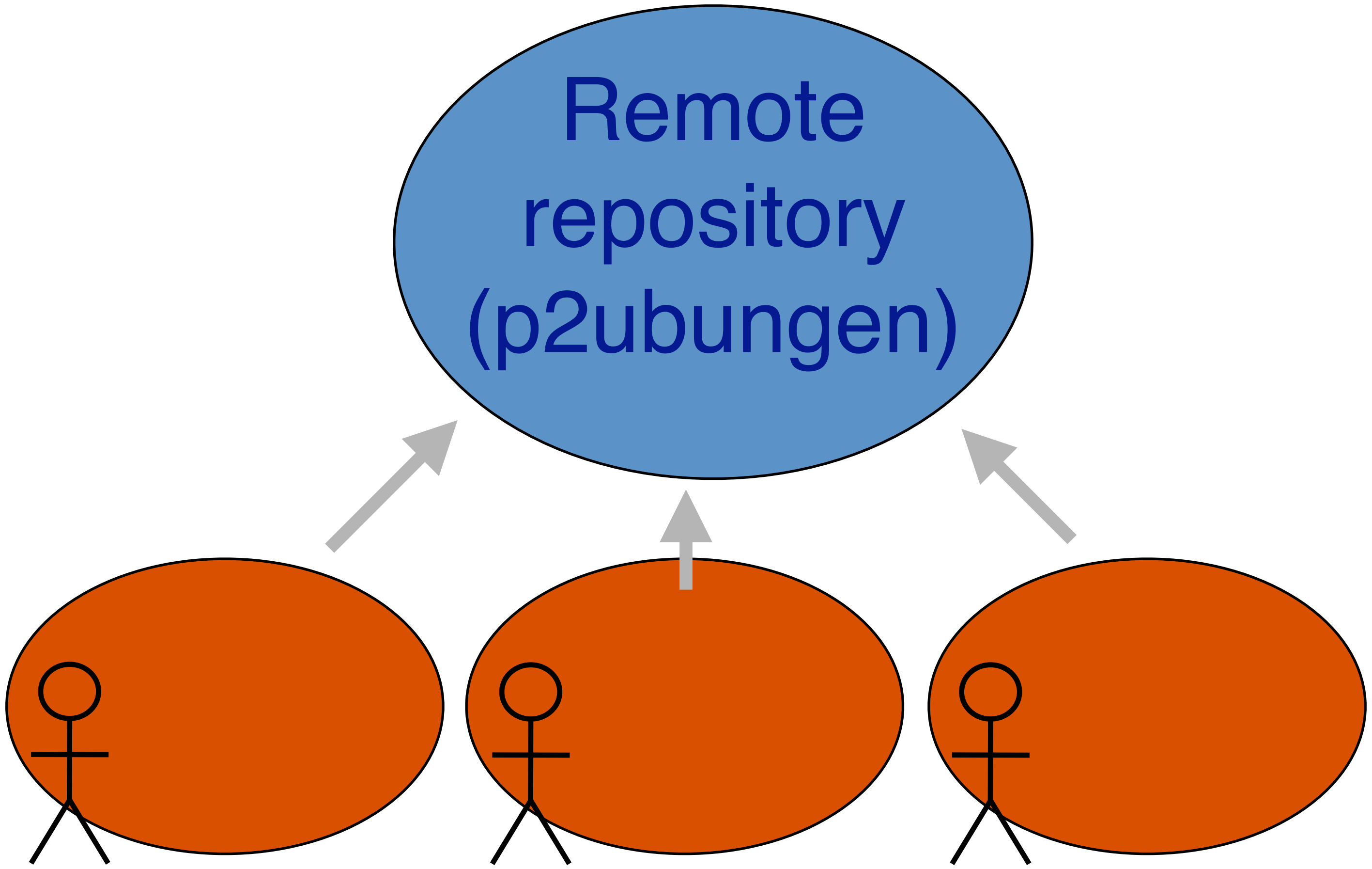


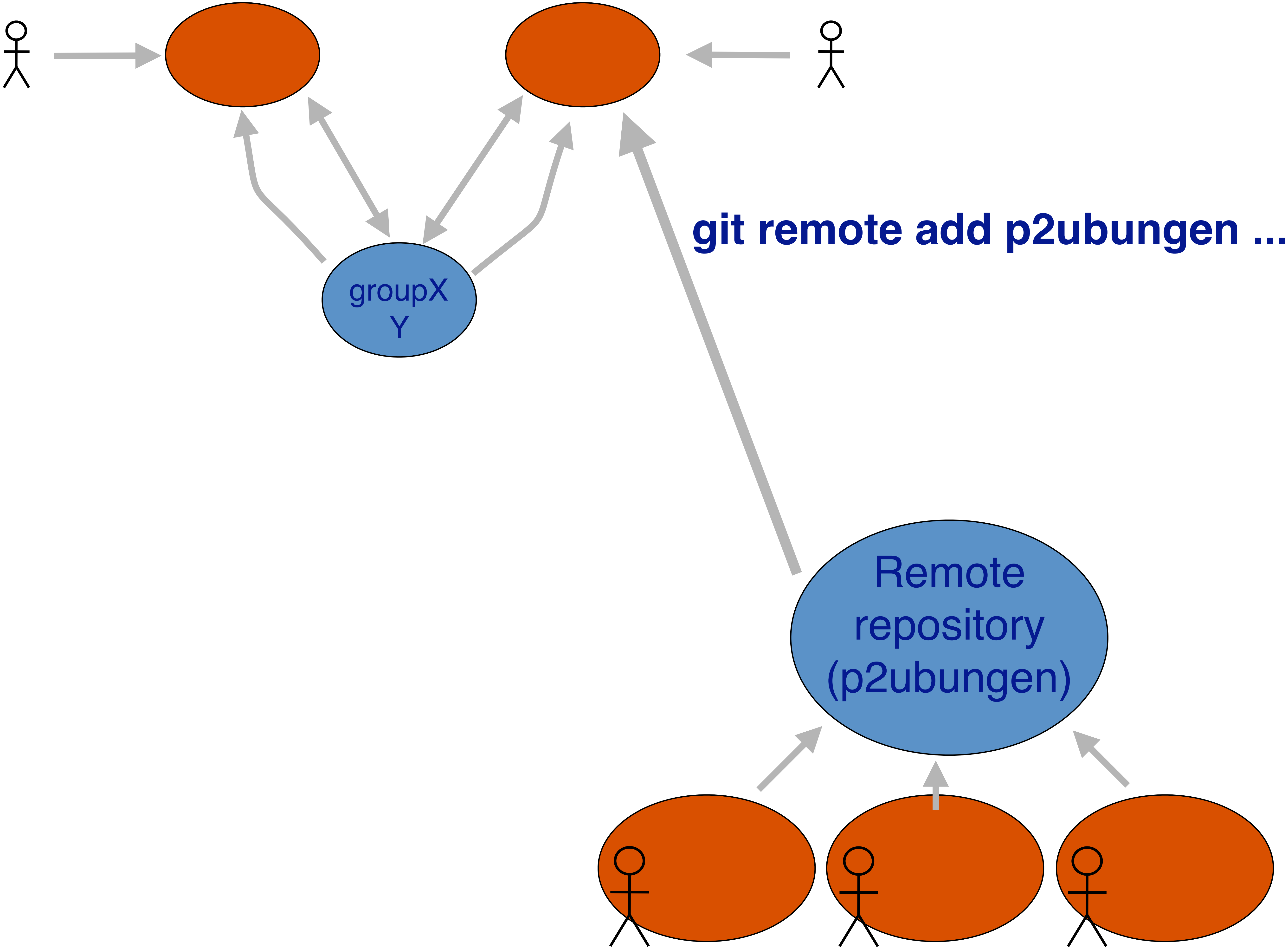
groupX



Remote repository (p2ubungen)


groupX

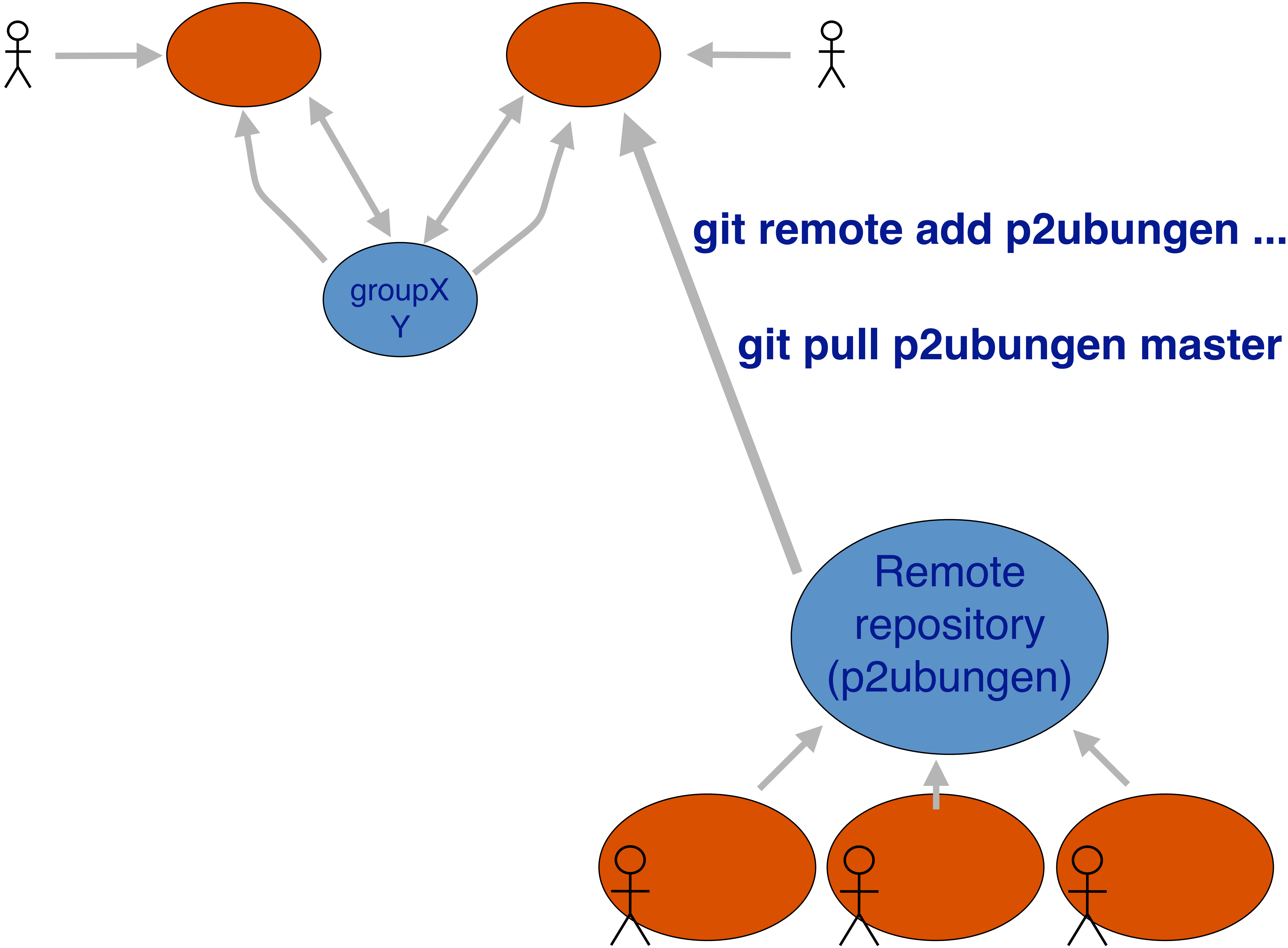

Remote repository (p2ubungen)



groupX Y

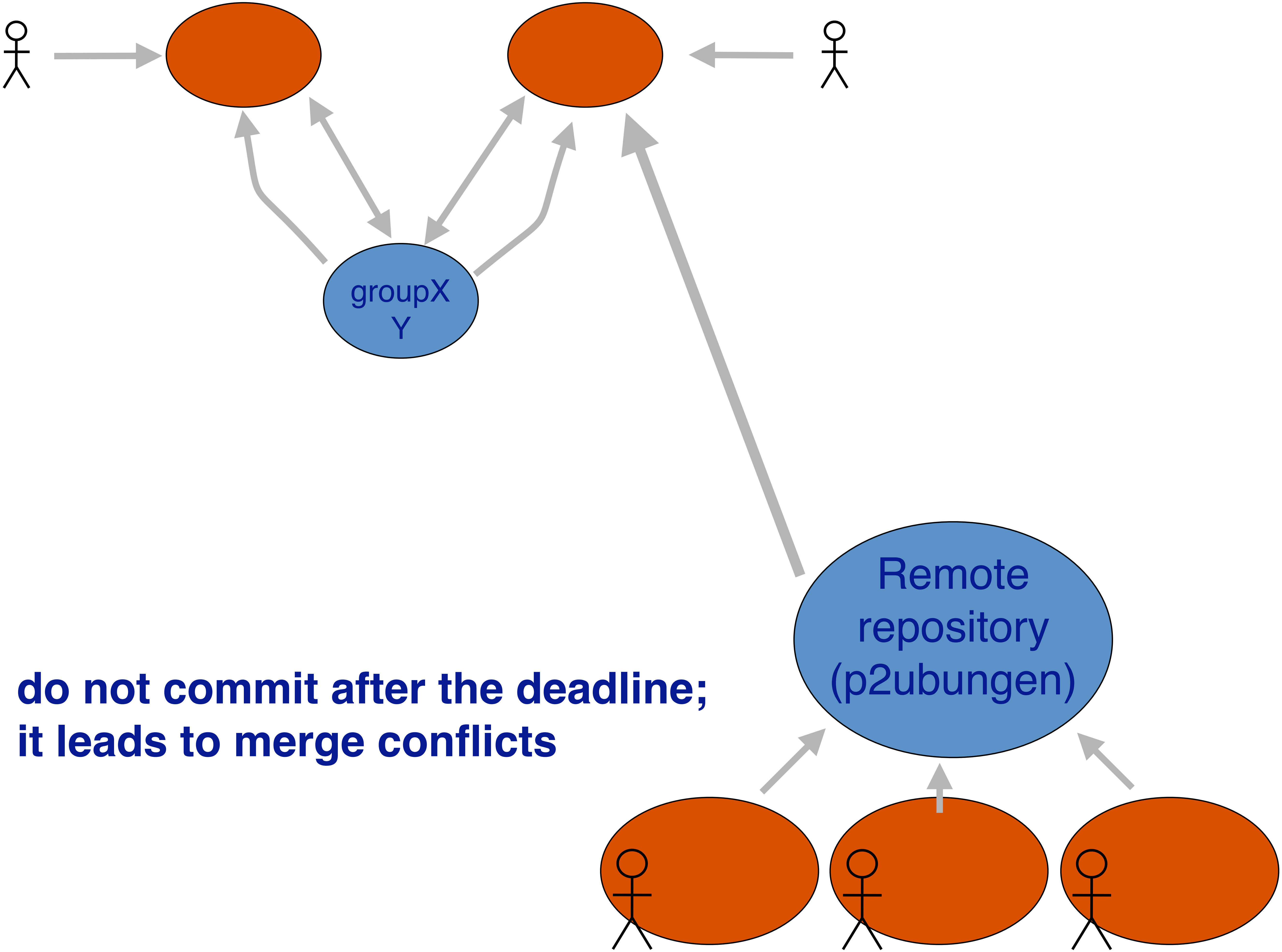
git remote add p2ubungen ...

Remote repository (p2ubungen)



```
git remote add p2ubungen ...
```

```
git pull p2ubungen master
```



groupX
Y

Remote repository (p2ubungen)

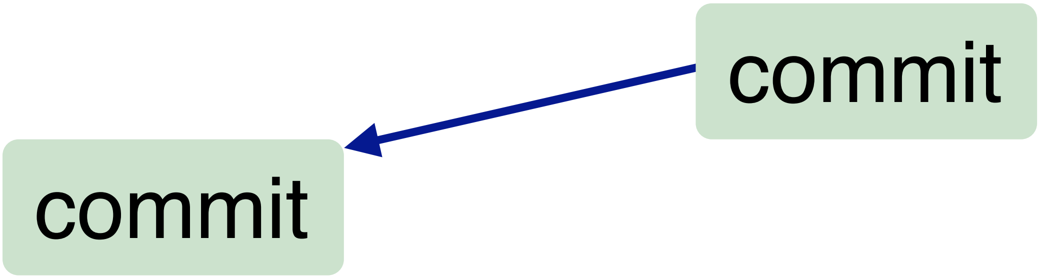
**do not commit after the deadline;
it leads to merge conflicts**

Basic git

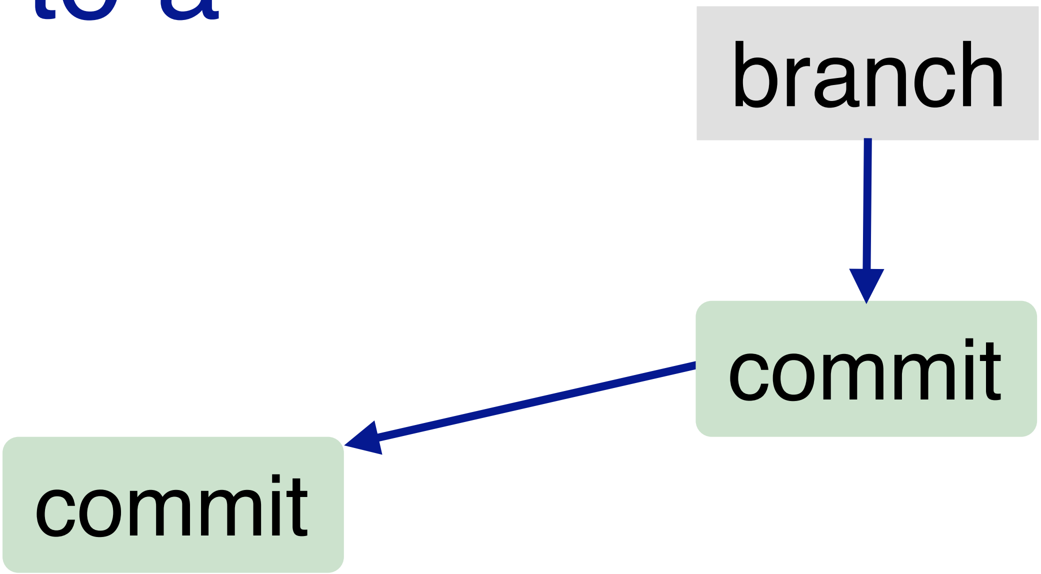
A “commit” is
“a set of changes”
to a “set of files”

commit

Most commits modify (or merge) earlier commits

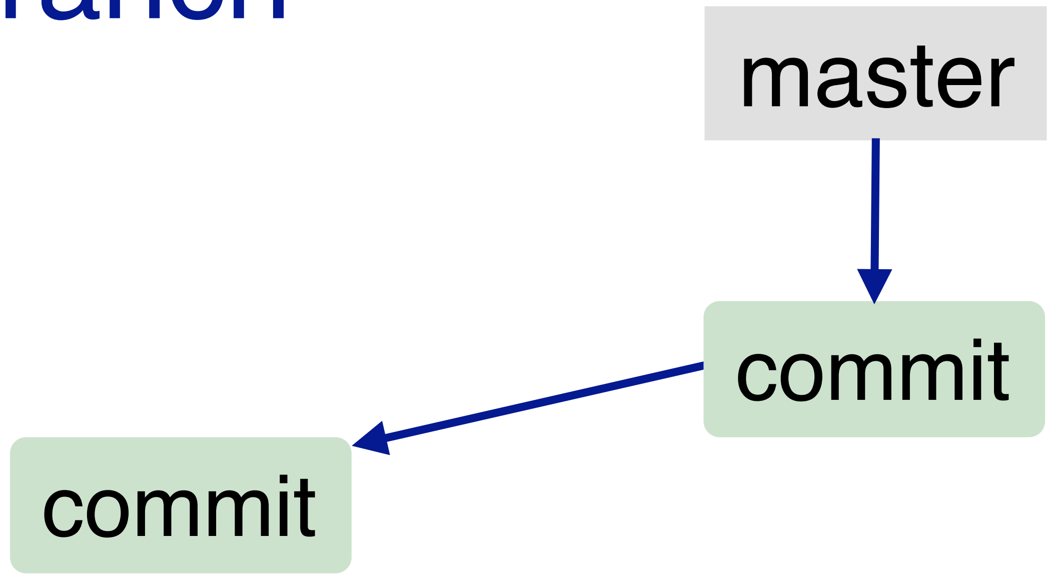


A graph of commits may belong to a *branch*

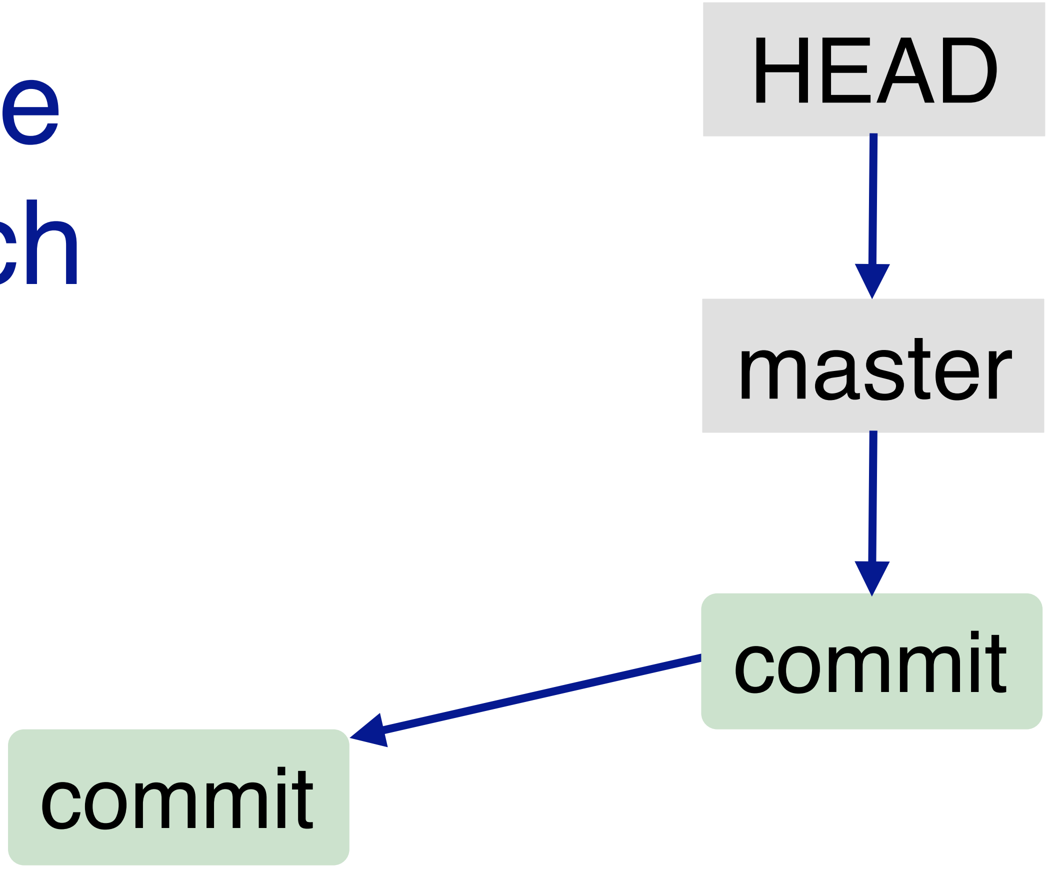


master

is the main branch

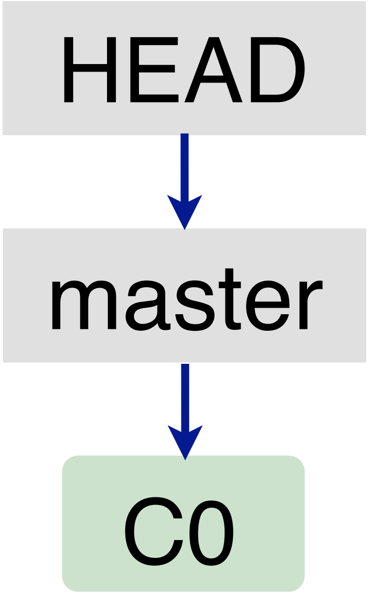


“HEAD “is the current branch

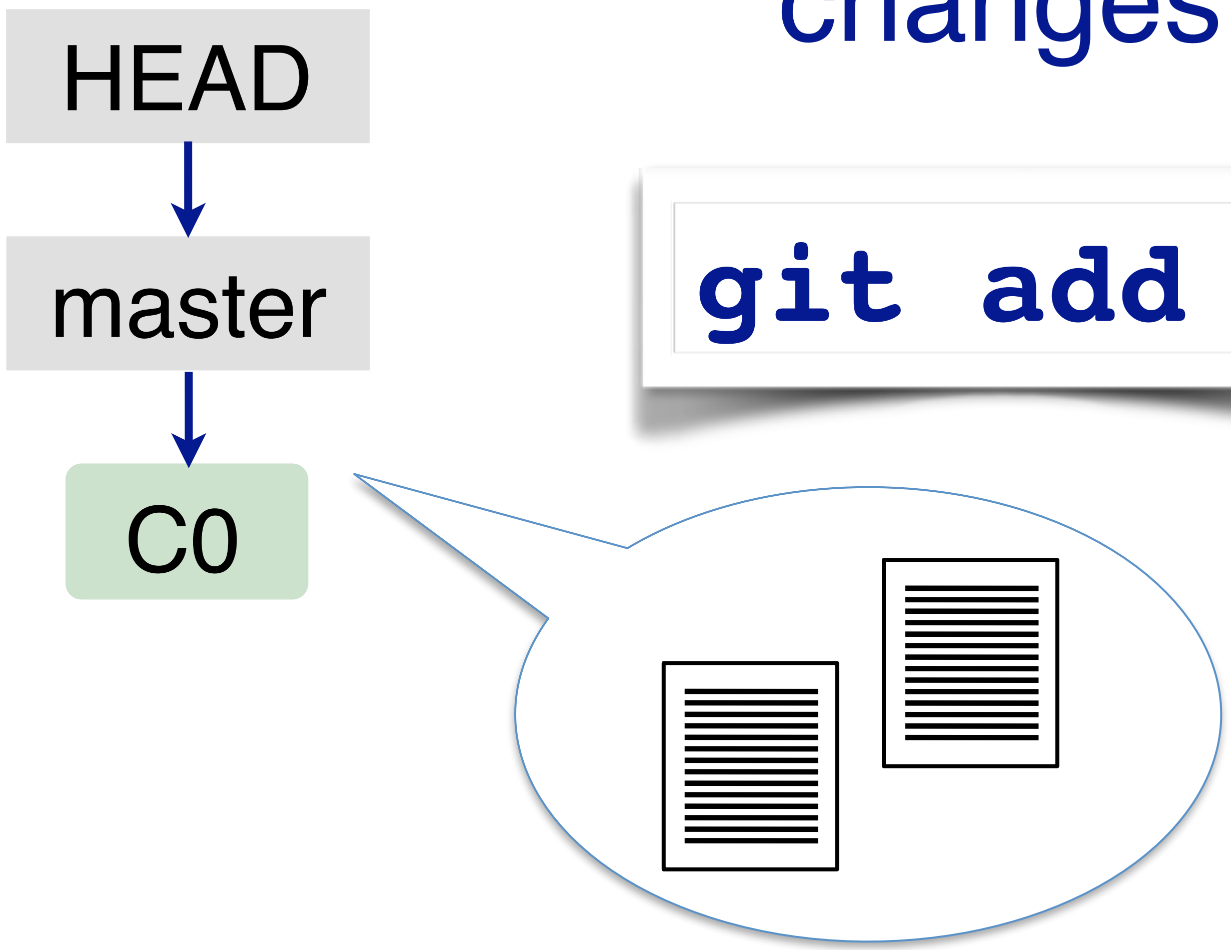


Create a git repo

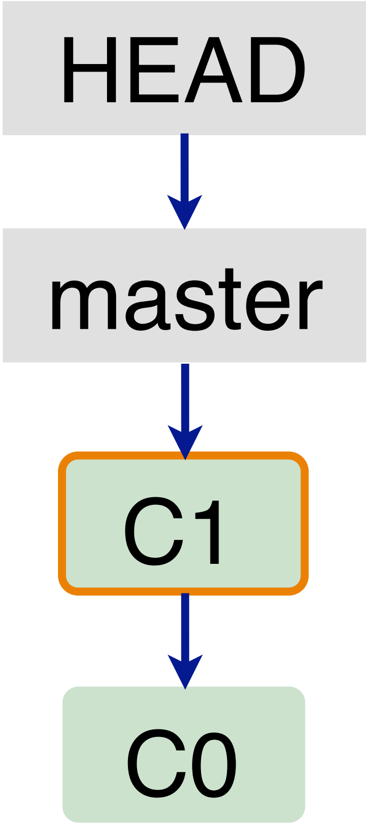
```
mkdir repo
cd repo
git init
```



Tell git to “stage” changes



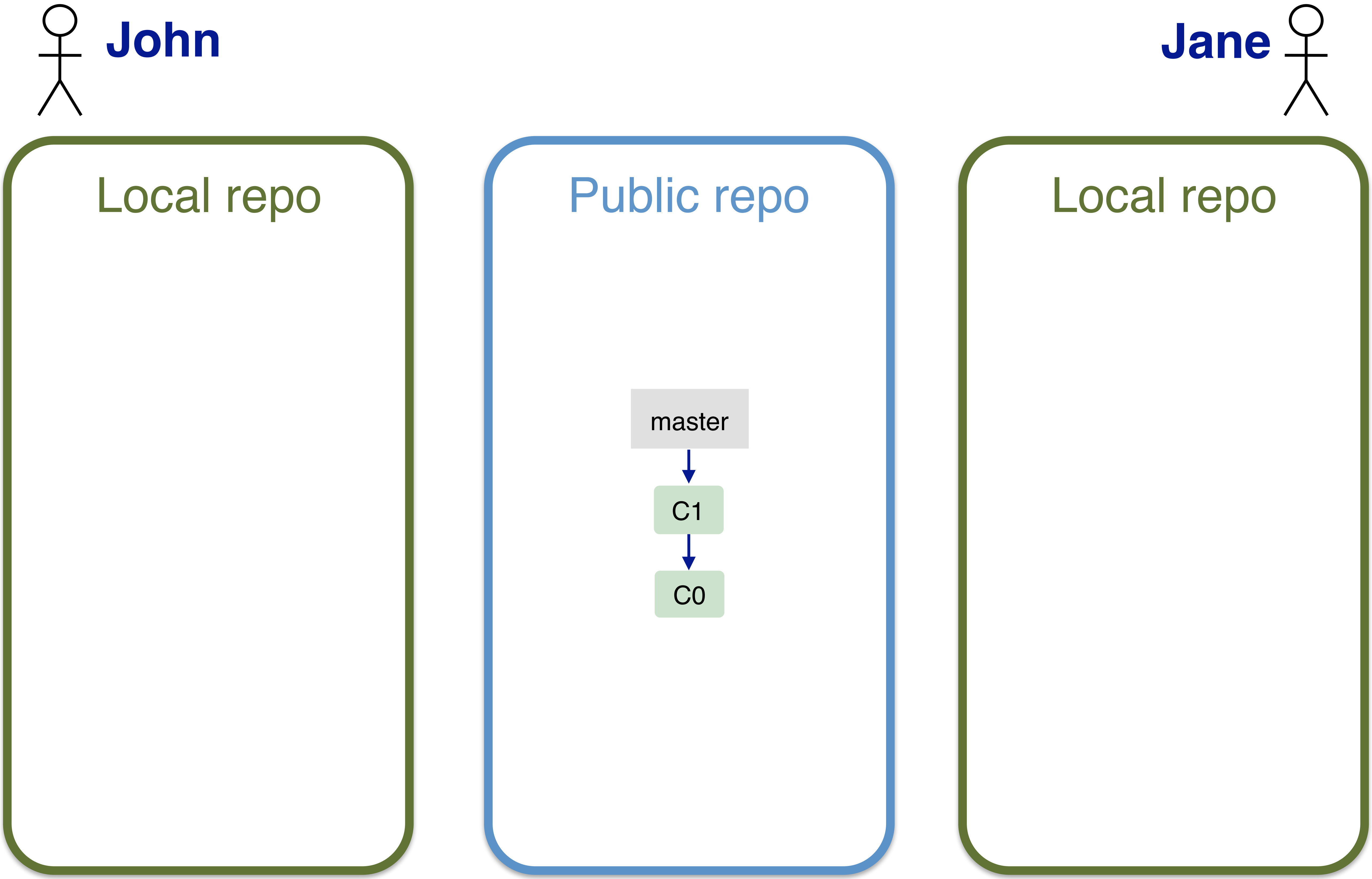
```
git add …
```



Commit your changes

```
git commit …
```

Collaborating



John

Jane

Local repo

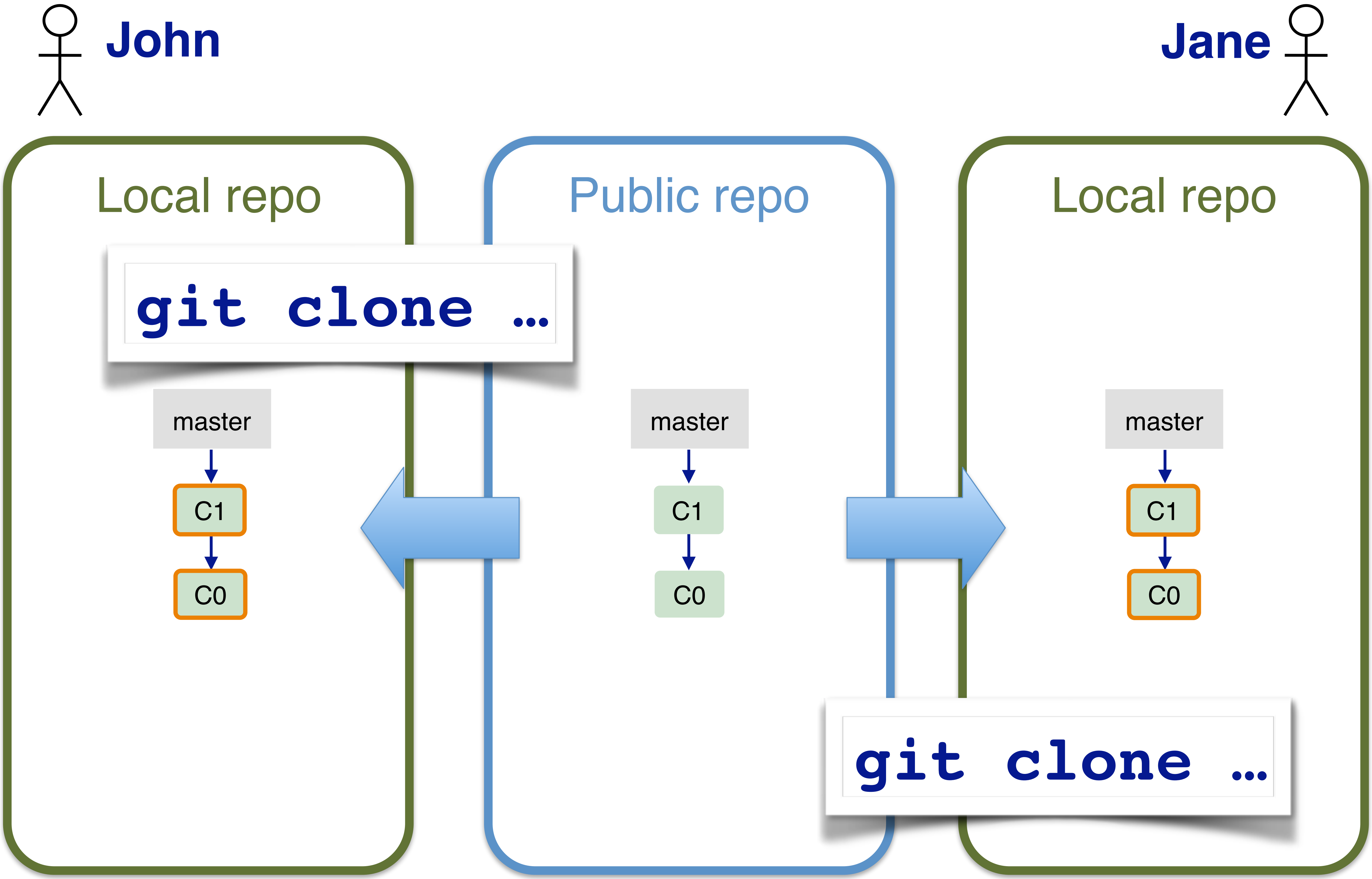
Public repo

Local repo

master

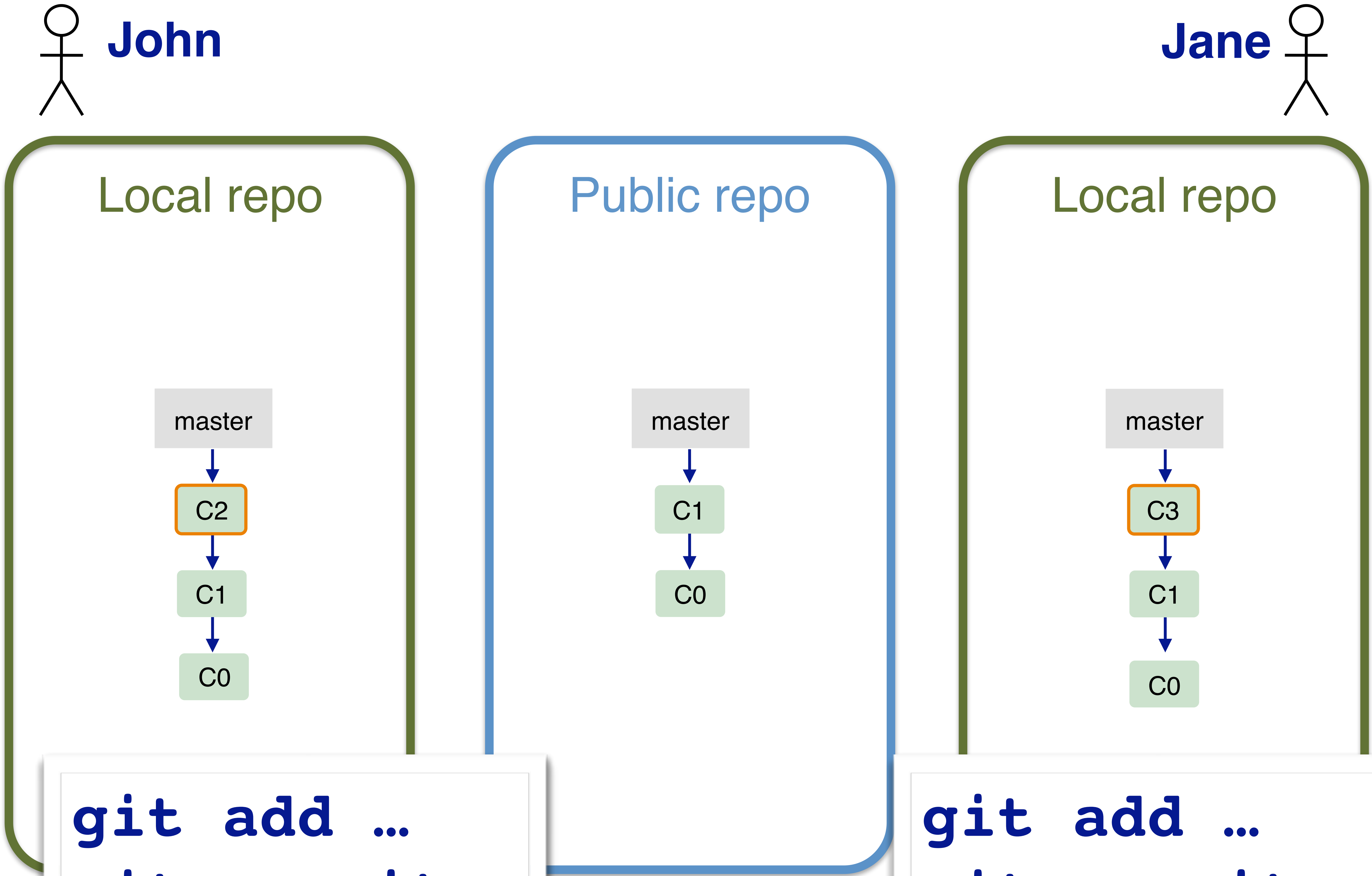
C1

C0



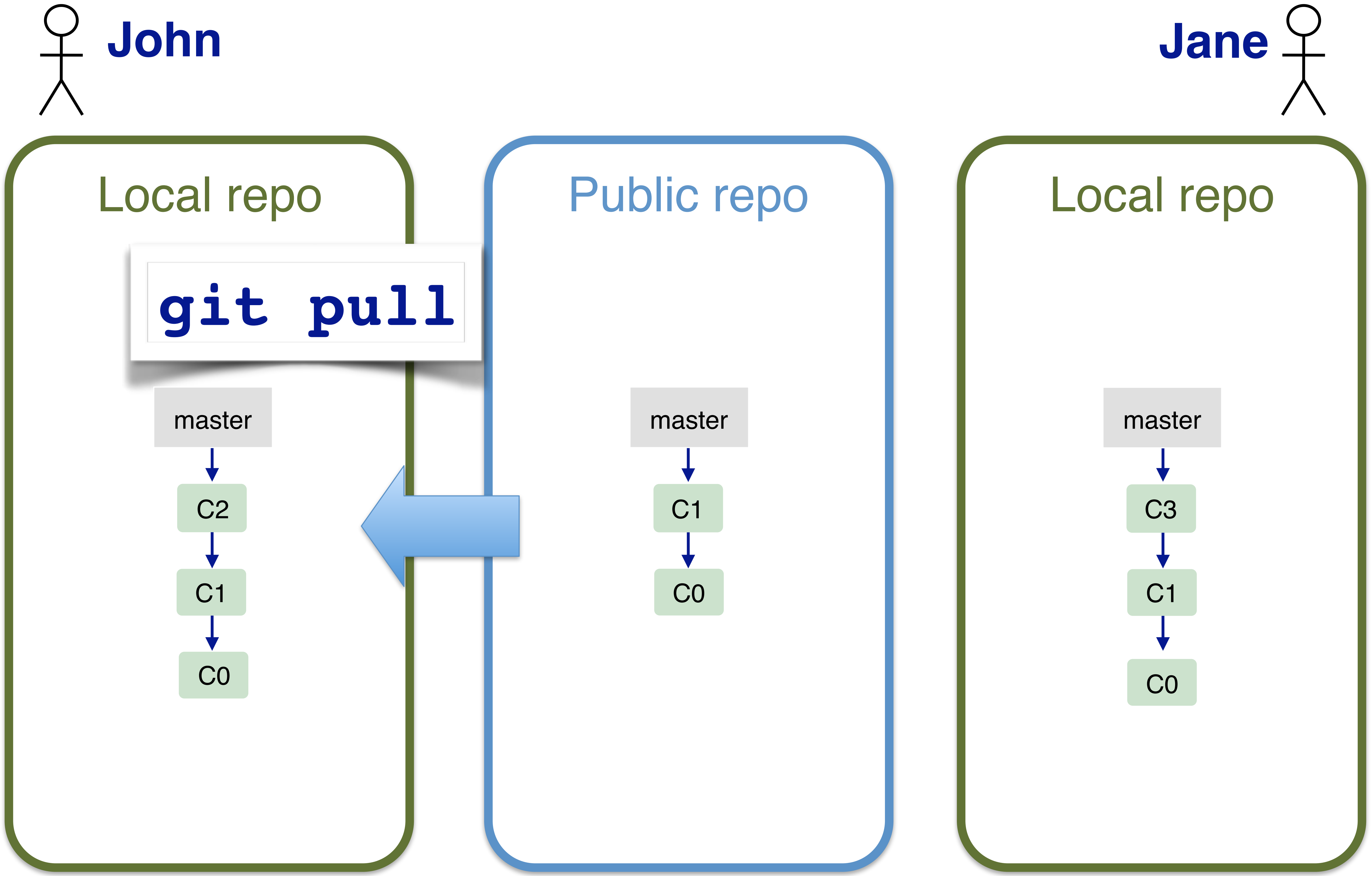
```
git clone …
```

```
git clone …
```



```
git add …
git commit …
```

```
git add …
git commit …
```



John

Jane

Local repo

`git pull`

master

C2

C1

C0

Public repo

master

C1

C0

Local repo

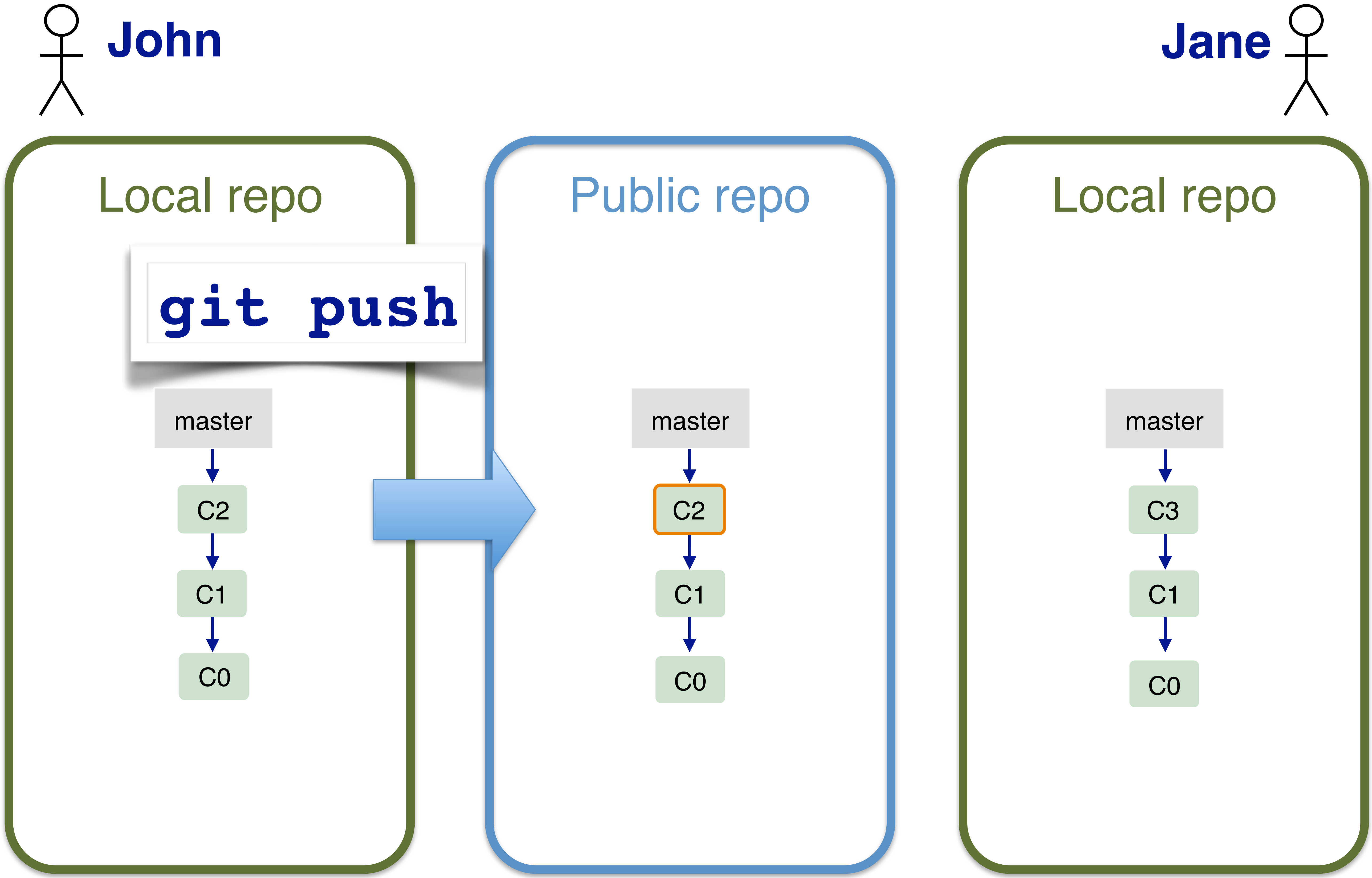
master

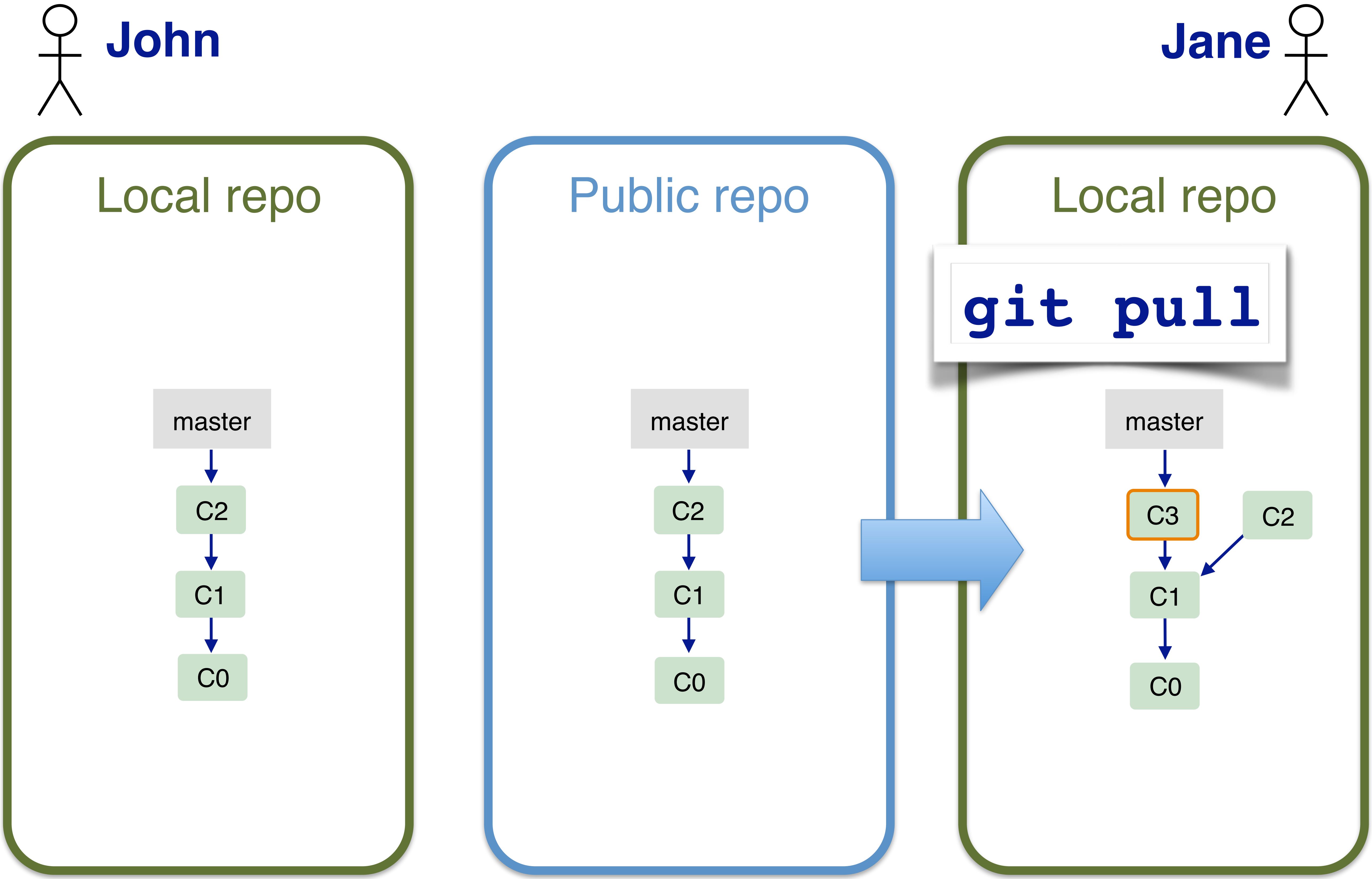
C3

C1

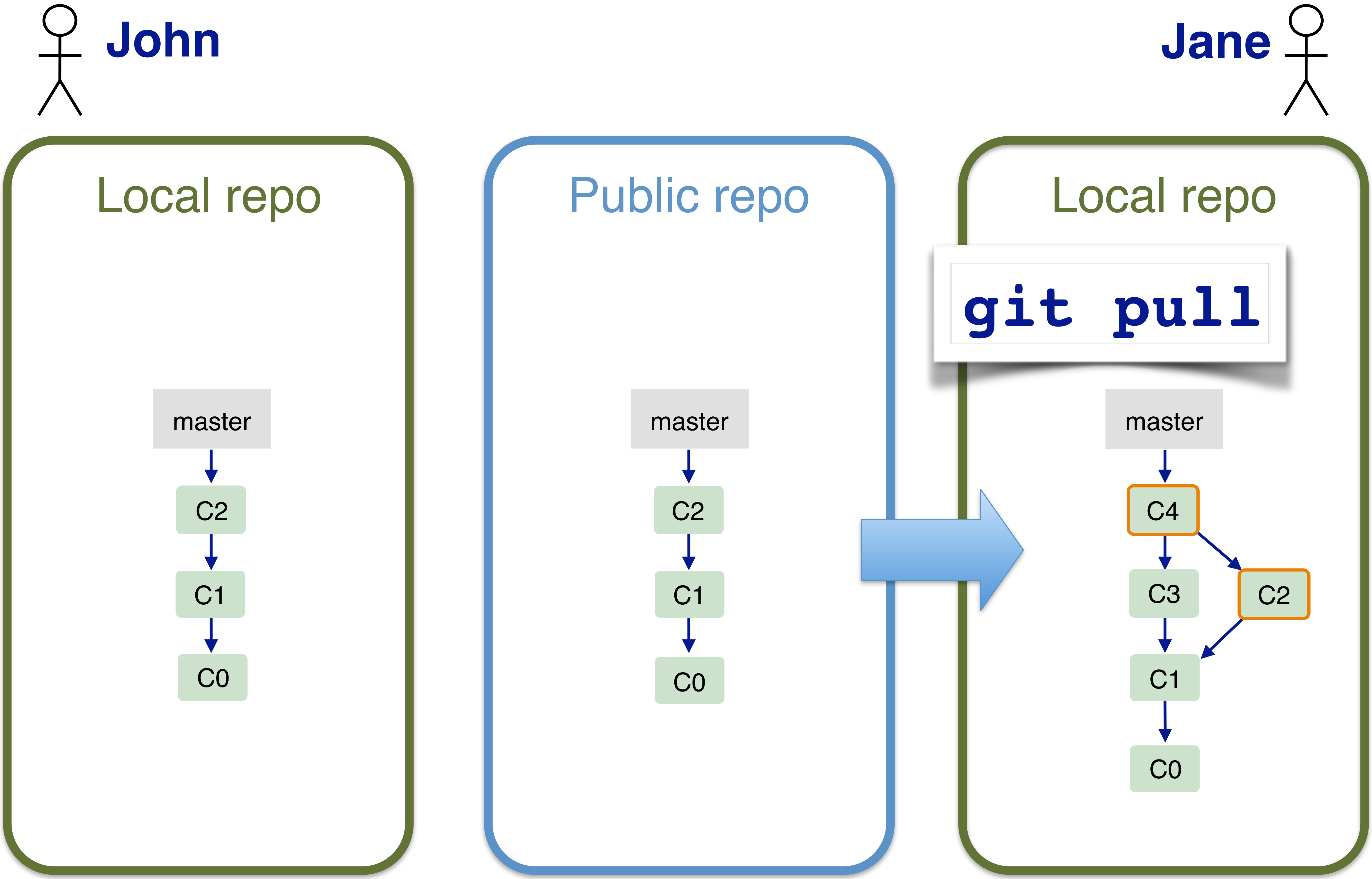
C0

(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

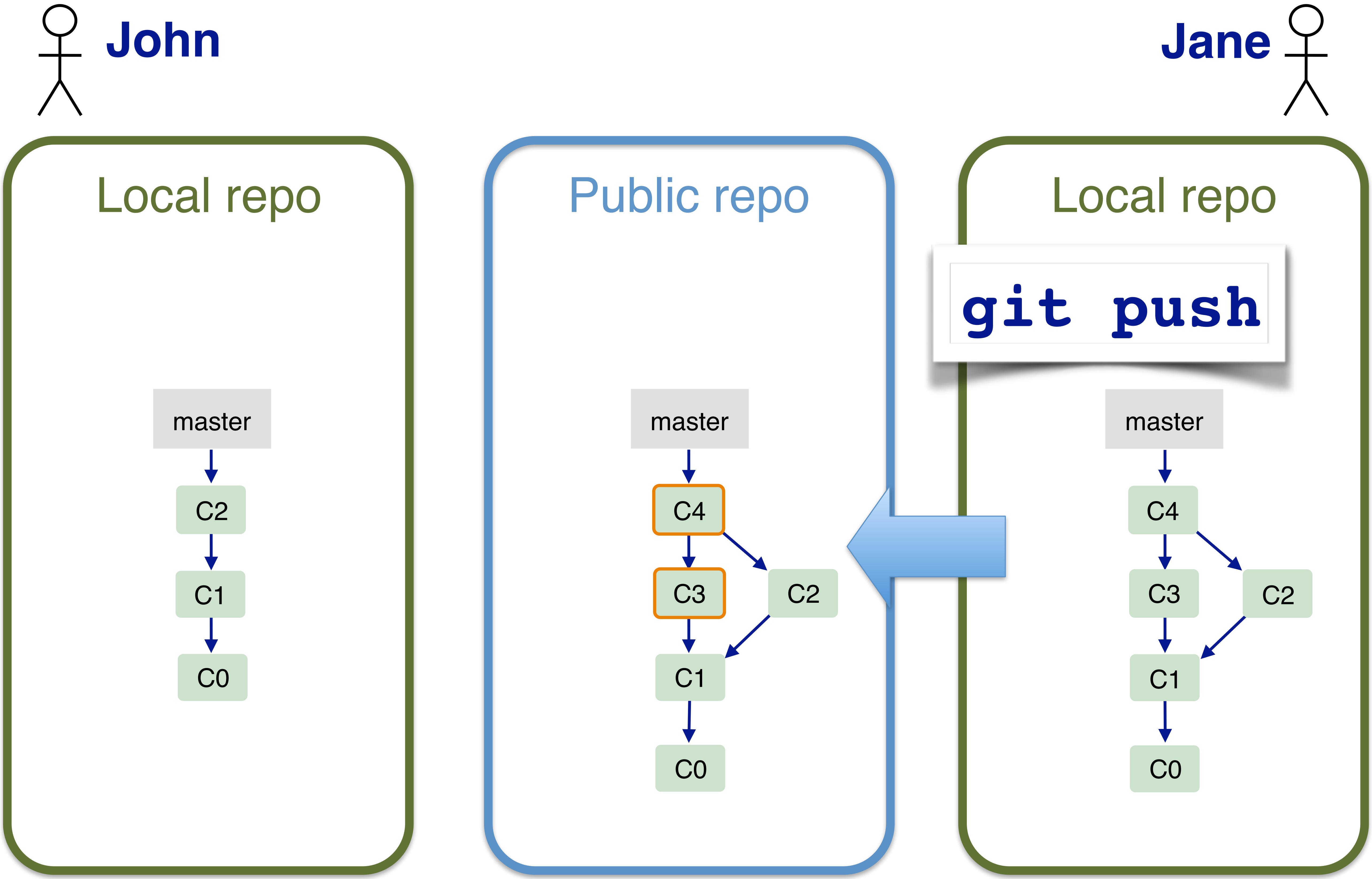

John
Jane
Local repo
master
C2
C1
C0
Public repo
master
C2
C1
C0
Local repo
master
C3
C2
C1
C0

```
git pull
```



```
git pull
```

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



John

Jane

Local repo

master

C2

C1

C0

Public repo

master

C4

C3

C2

C1

C0

Local repo

```
git push
```

master

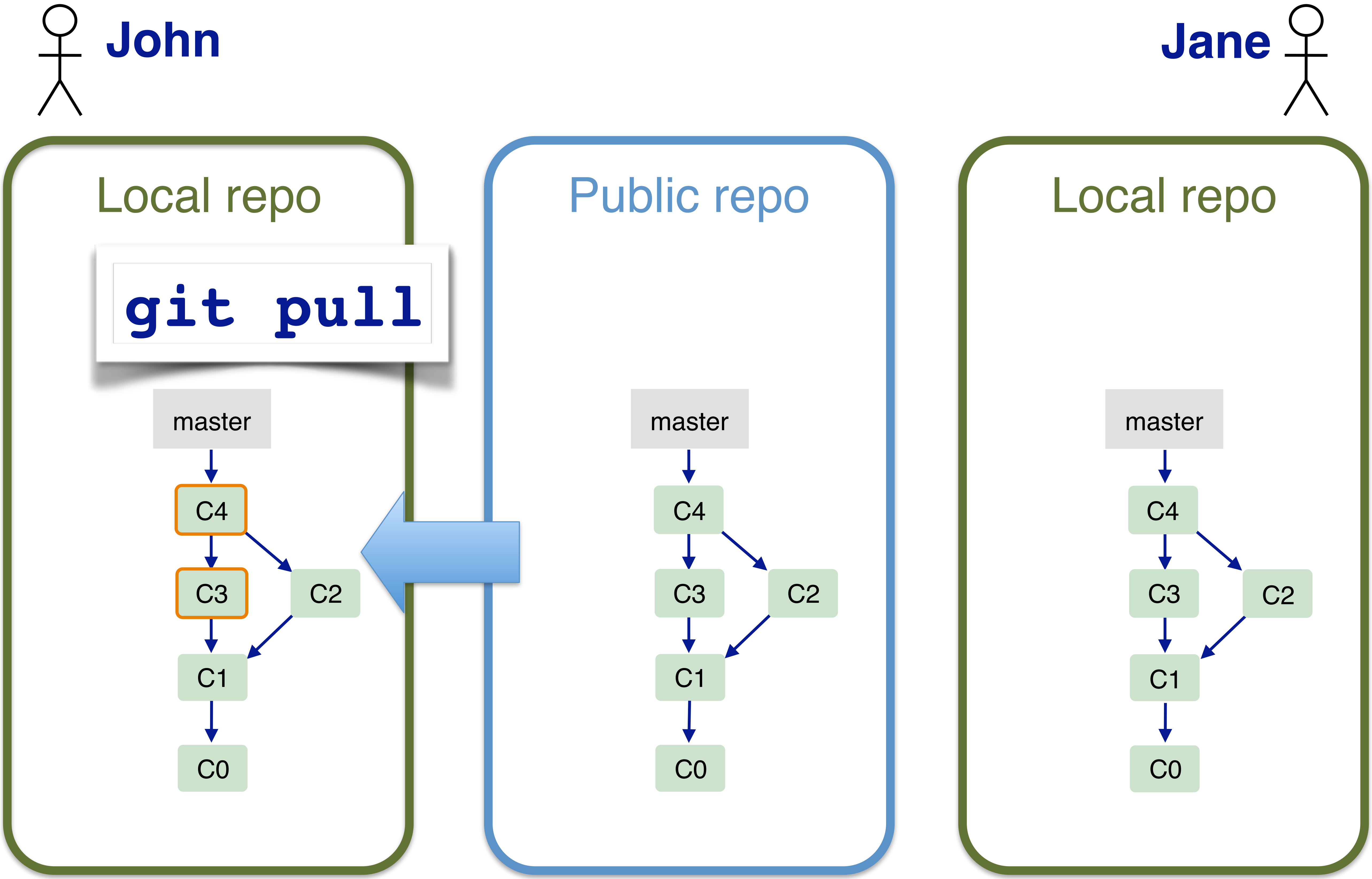
C4

C3

C2

C1

C0


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

```
git pull
```

to be continued

Resources

http://git-scm.com/

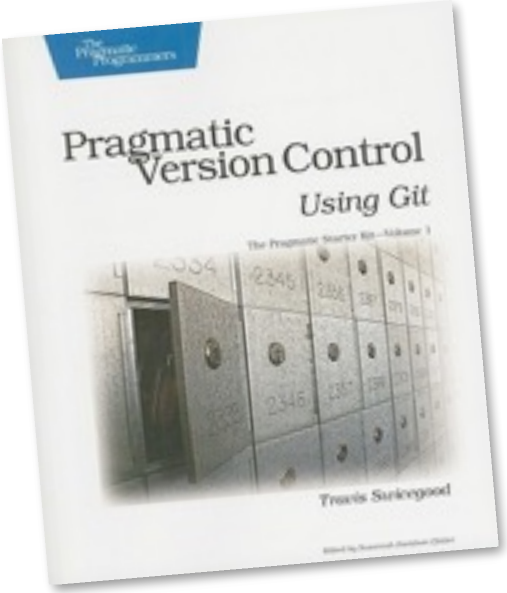
http://book.git-scm.com/index.html

https://github.com/

Getting Git

Scott Chacon

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



The Pragmatic Programmers

Pragmatic Version Control

Using Git

The Pragmatic Starter Kit—Volume 1

Travis Swicegood

http://oreilly.com/