UNIVERSITÄT
BERN

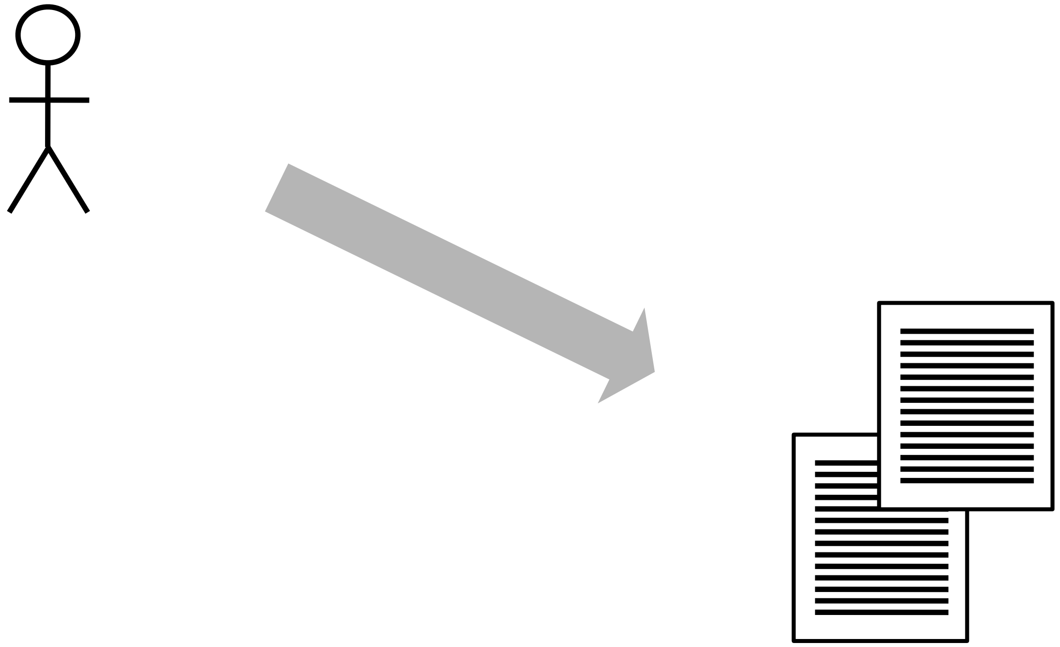
Distributed version control with git — a brief introduction

Andrei Chis

based on slides by
Oscar Nierstrasz

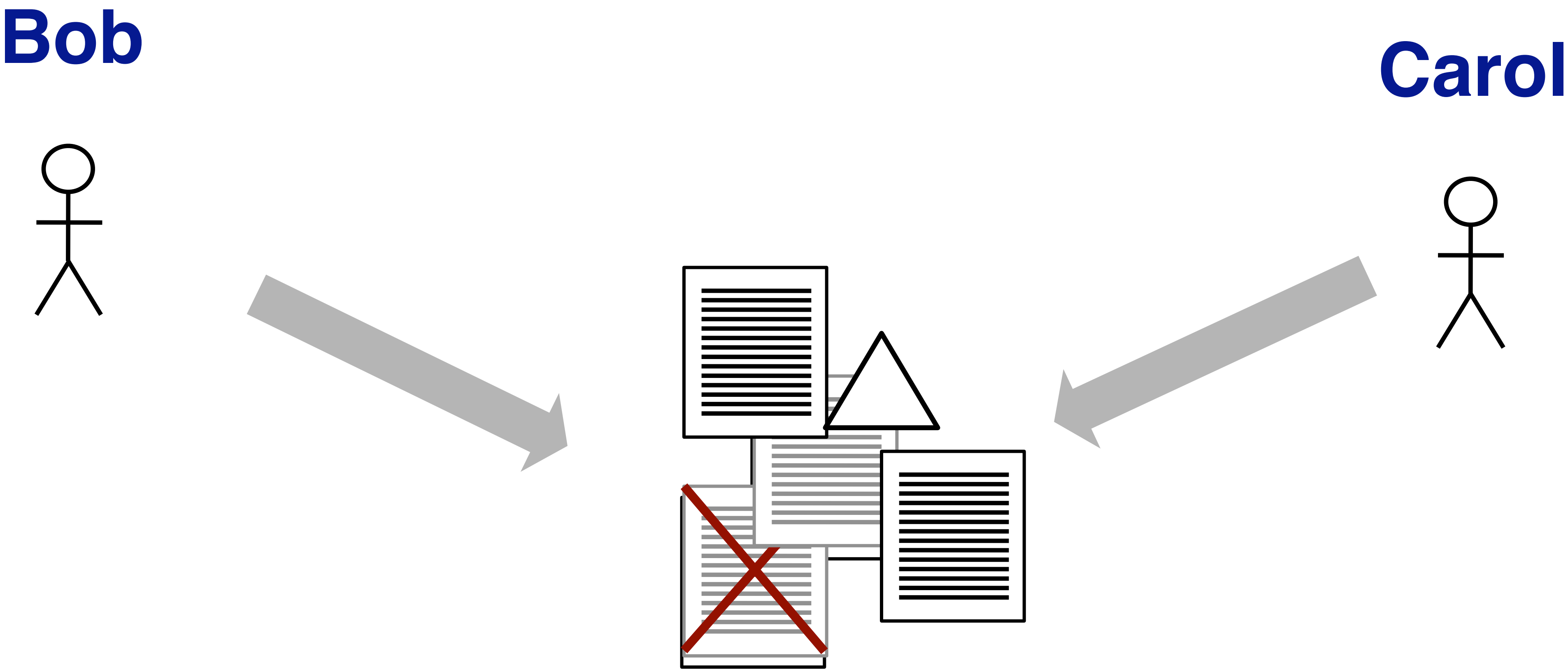
Why version control?

Why version control?



Bob

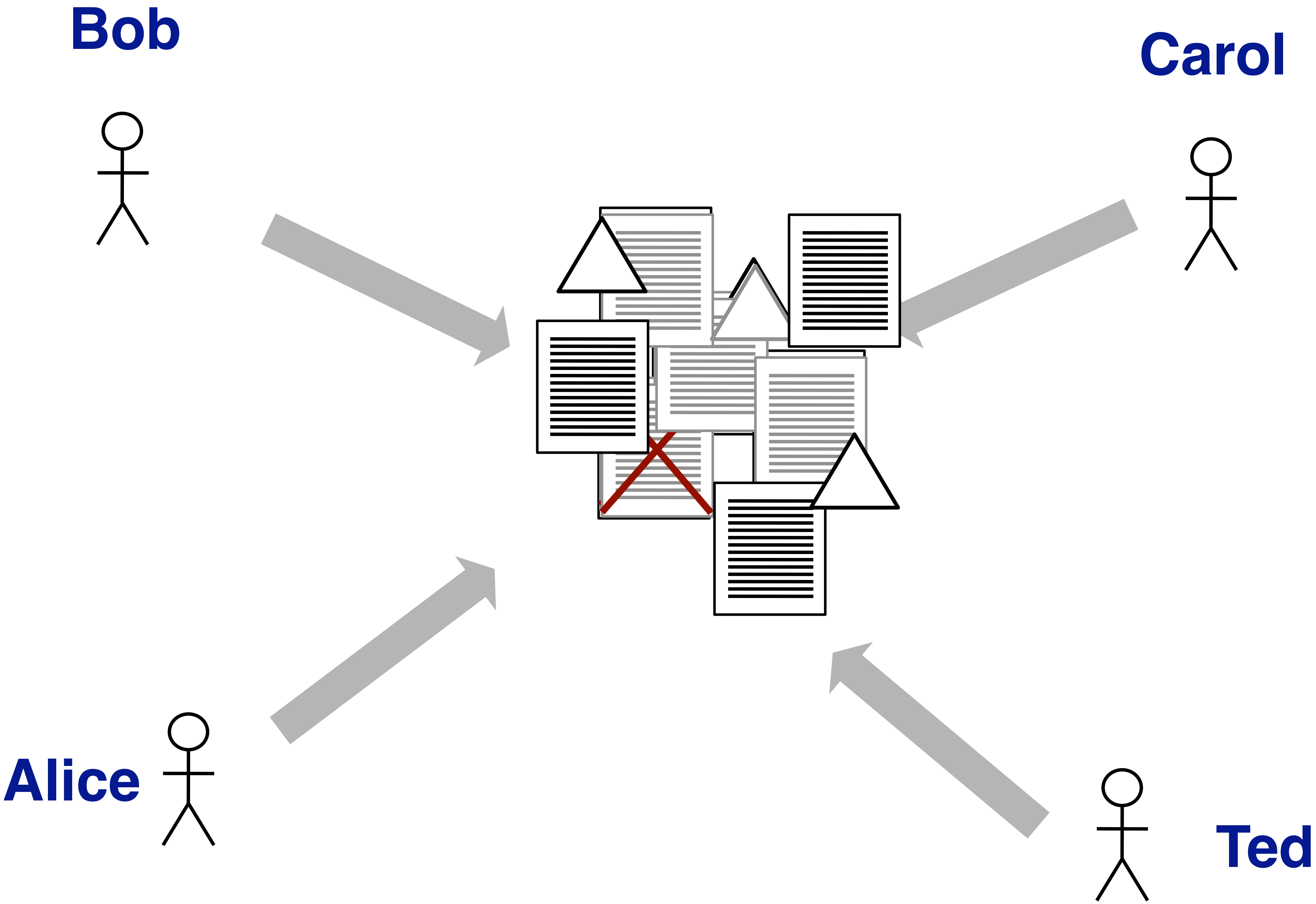
Why version control?



Bob

Carol

Why version control?



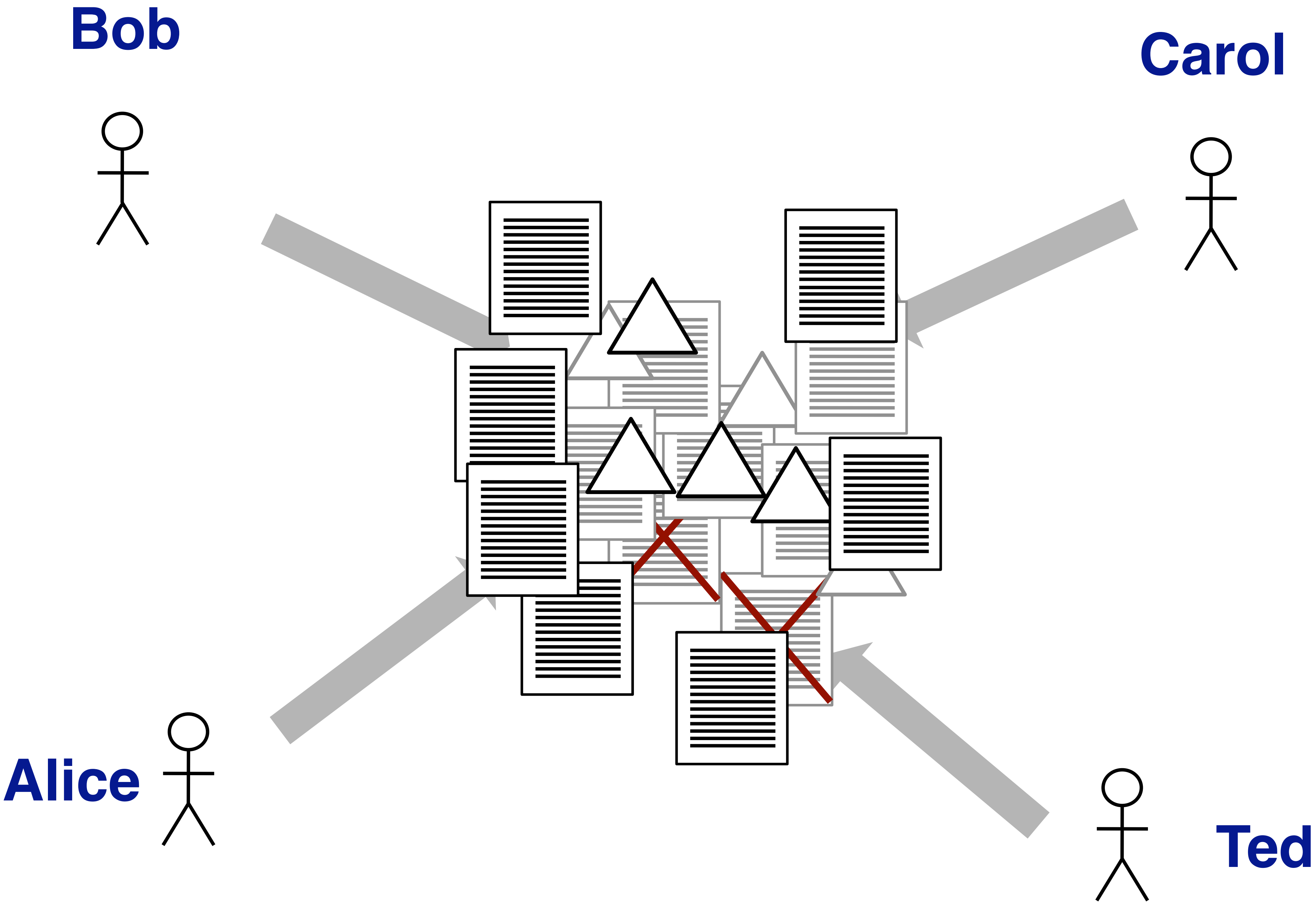
Bob

Carol

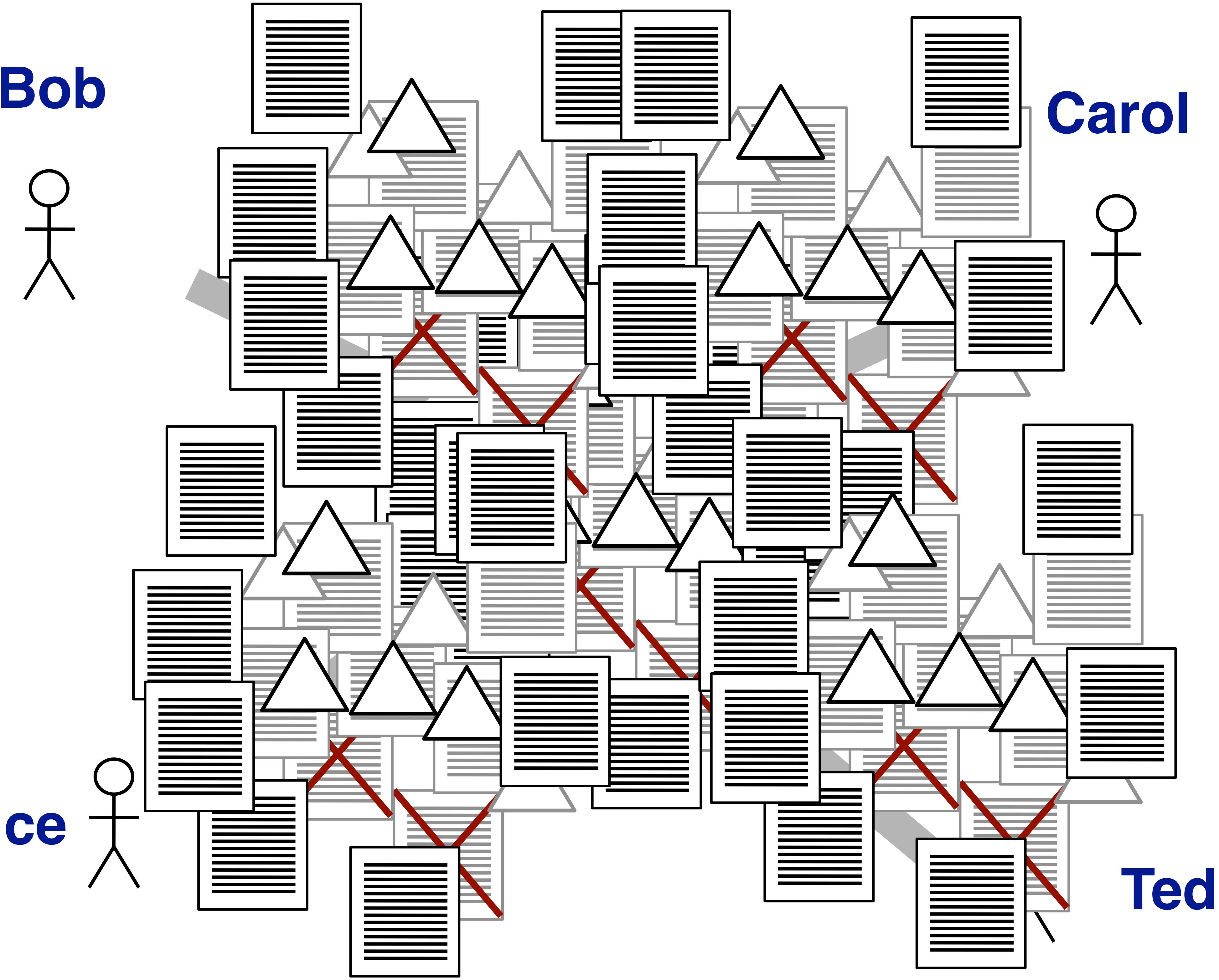
Alice

Ted

Why version control?



Why version control?



Bob

Carol

Alice

Ted

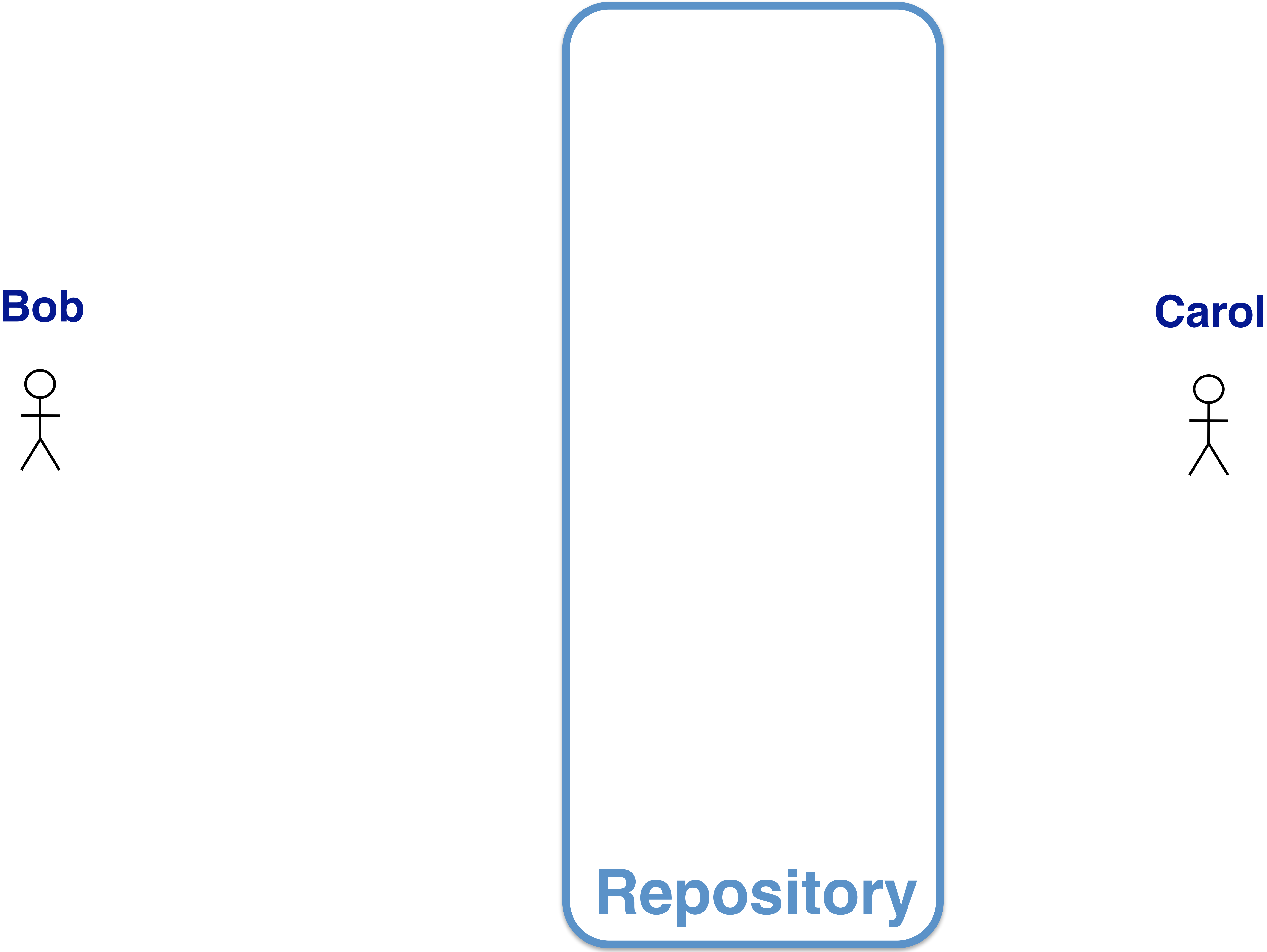
Why version control?

Bob
Carol
Alice
Ted

A recipe for disaster!

Why version control?

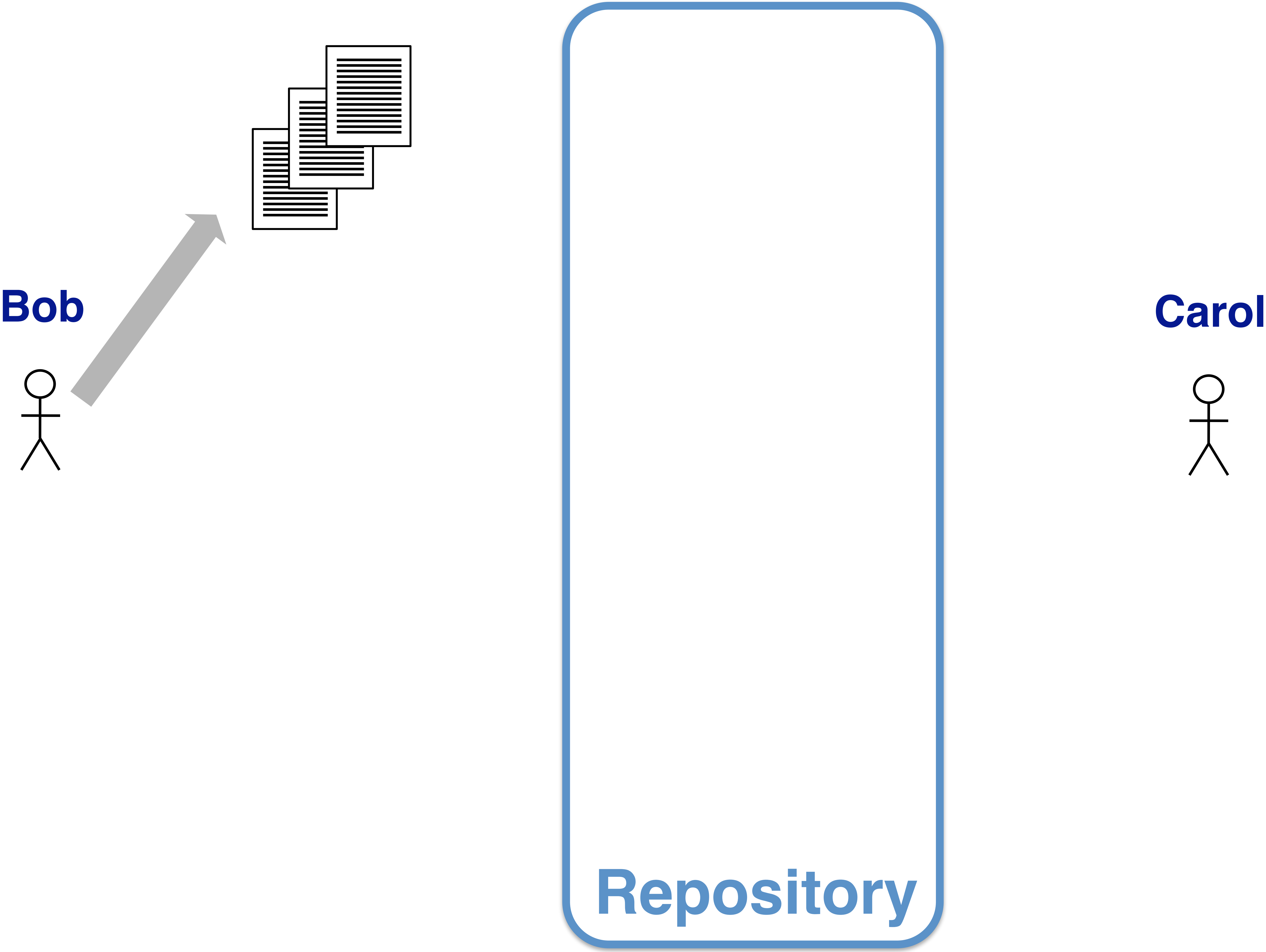
Cope with the confusion that happens when multiple people edit the same files



Bob

Repository

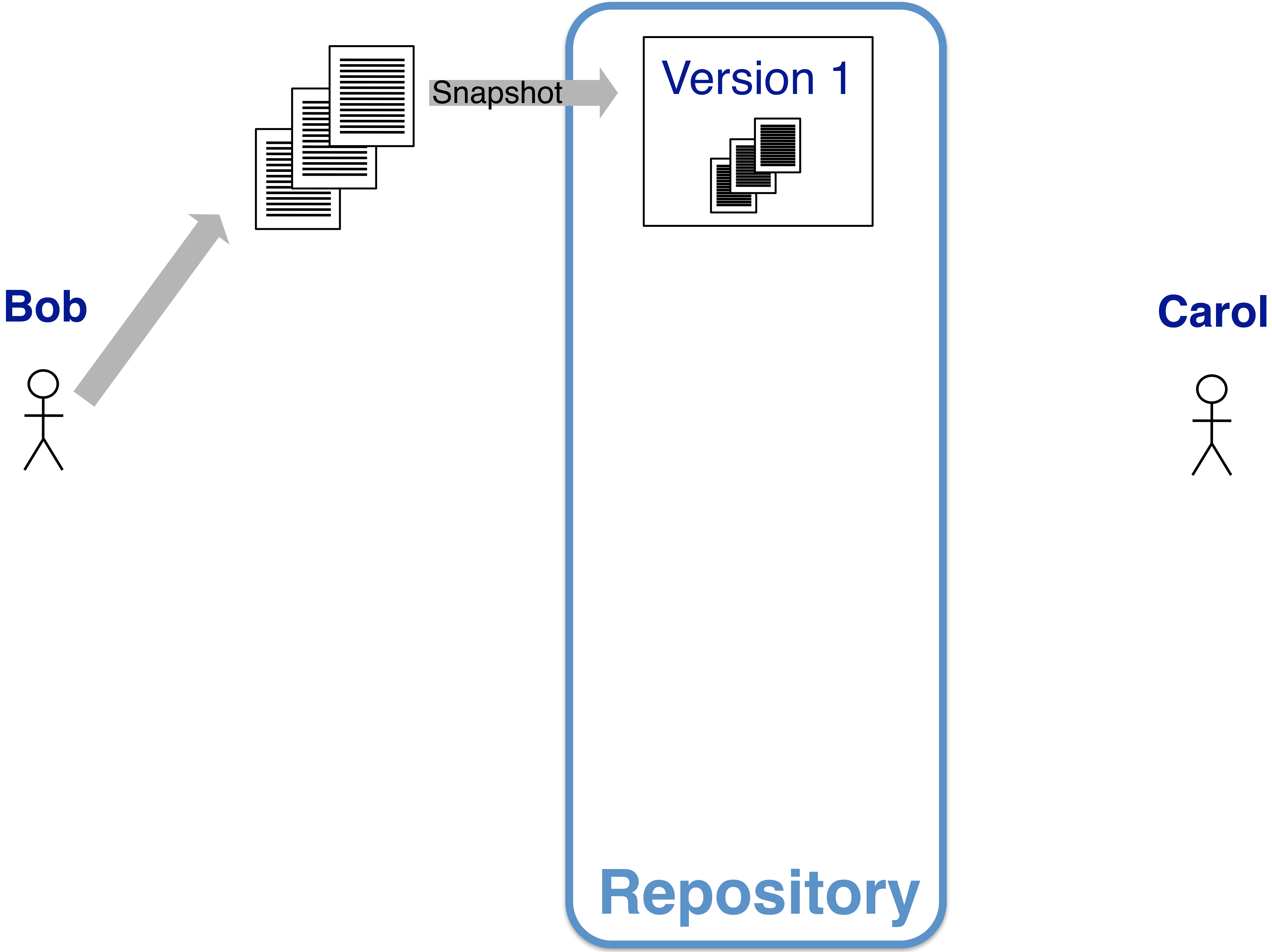
Carol



Bob

Repository

Carol



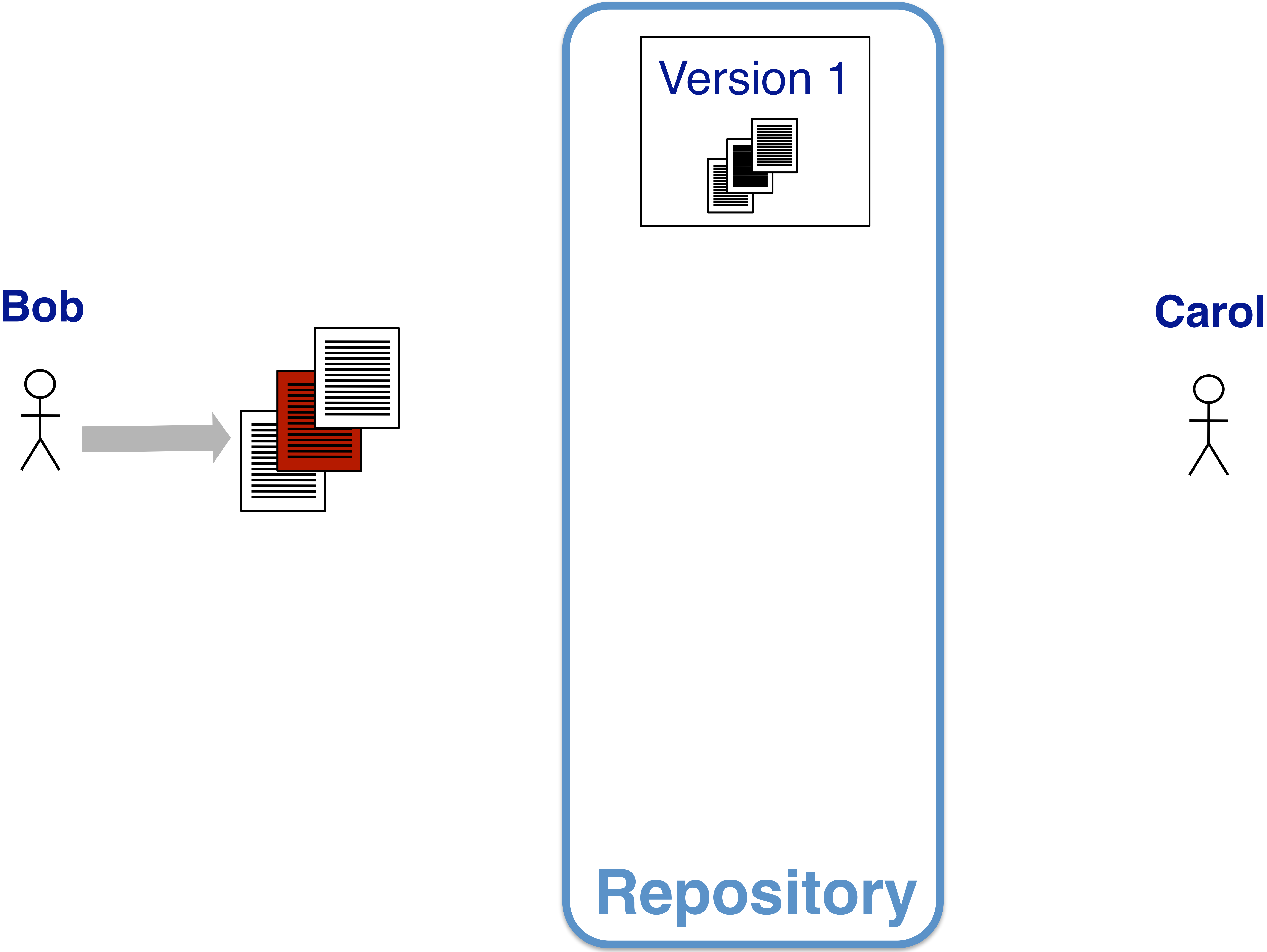
Bob

Snapshot

Version 1

Repository

Carol

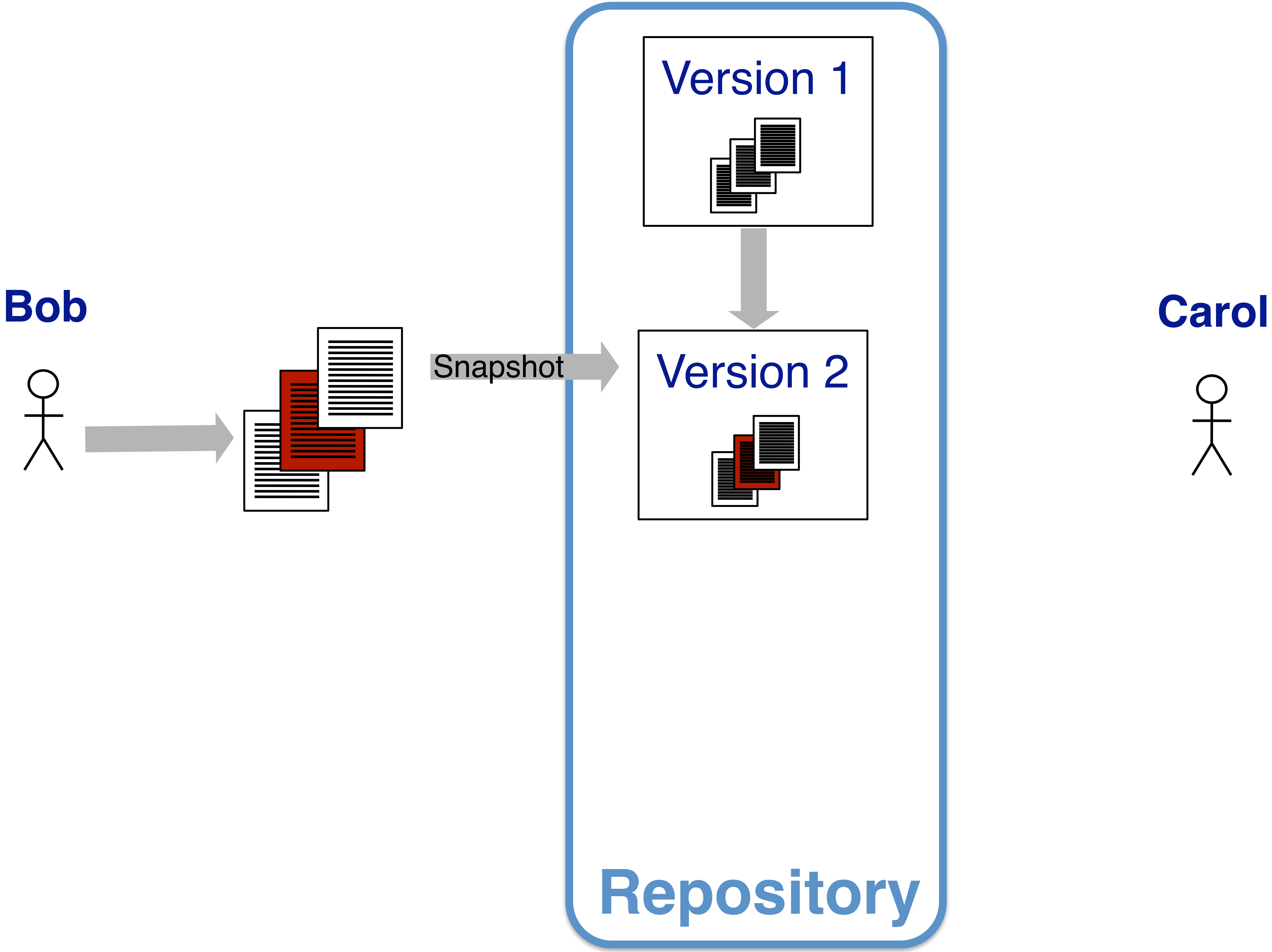


Bob

Version 1

Repository

Carol



Bob

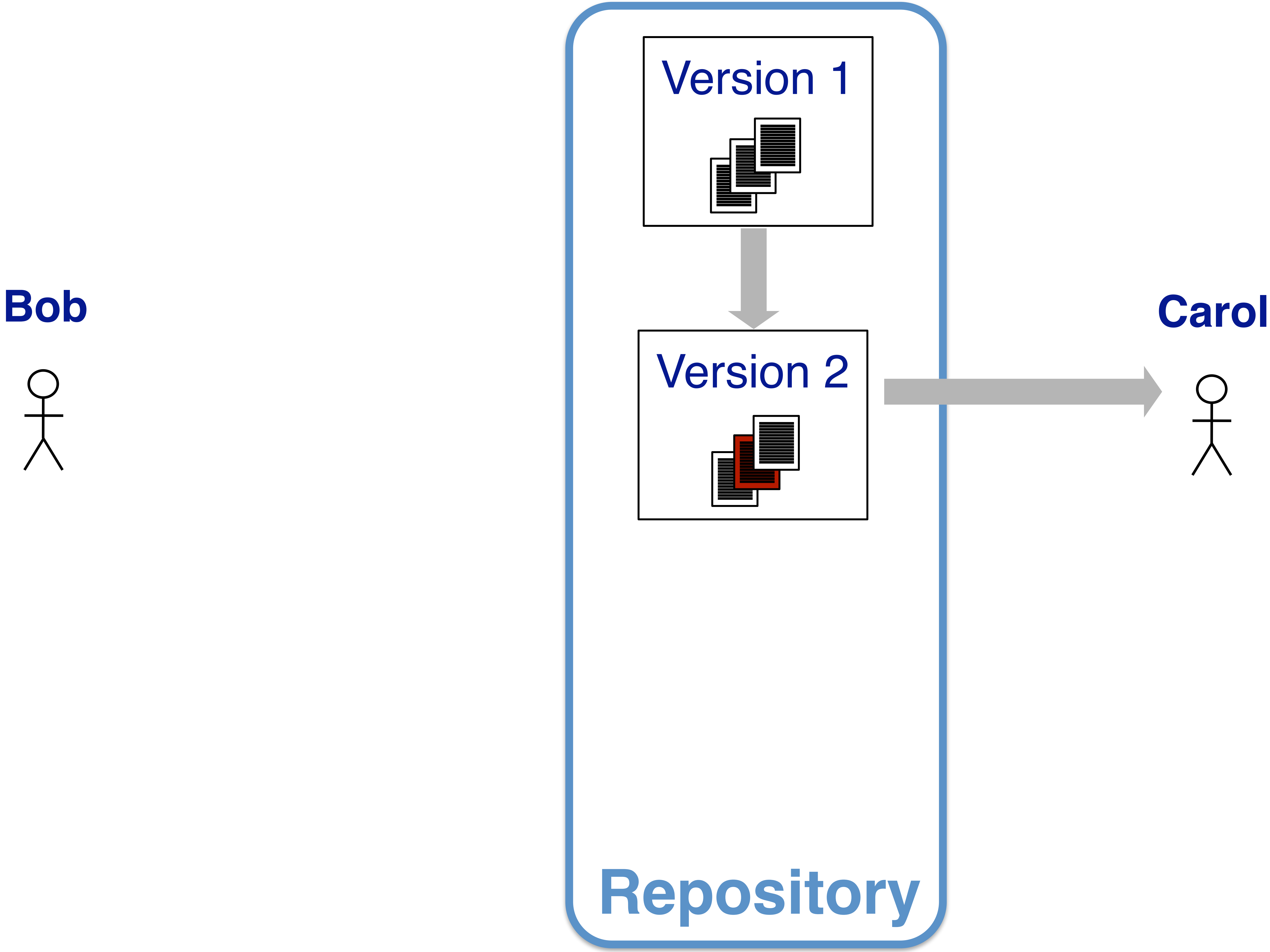
Snapshot

Version 1

Version 2

Repository

Carol



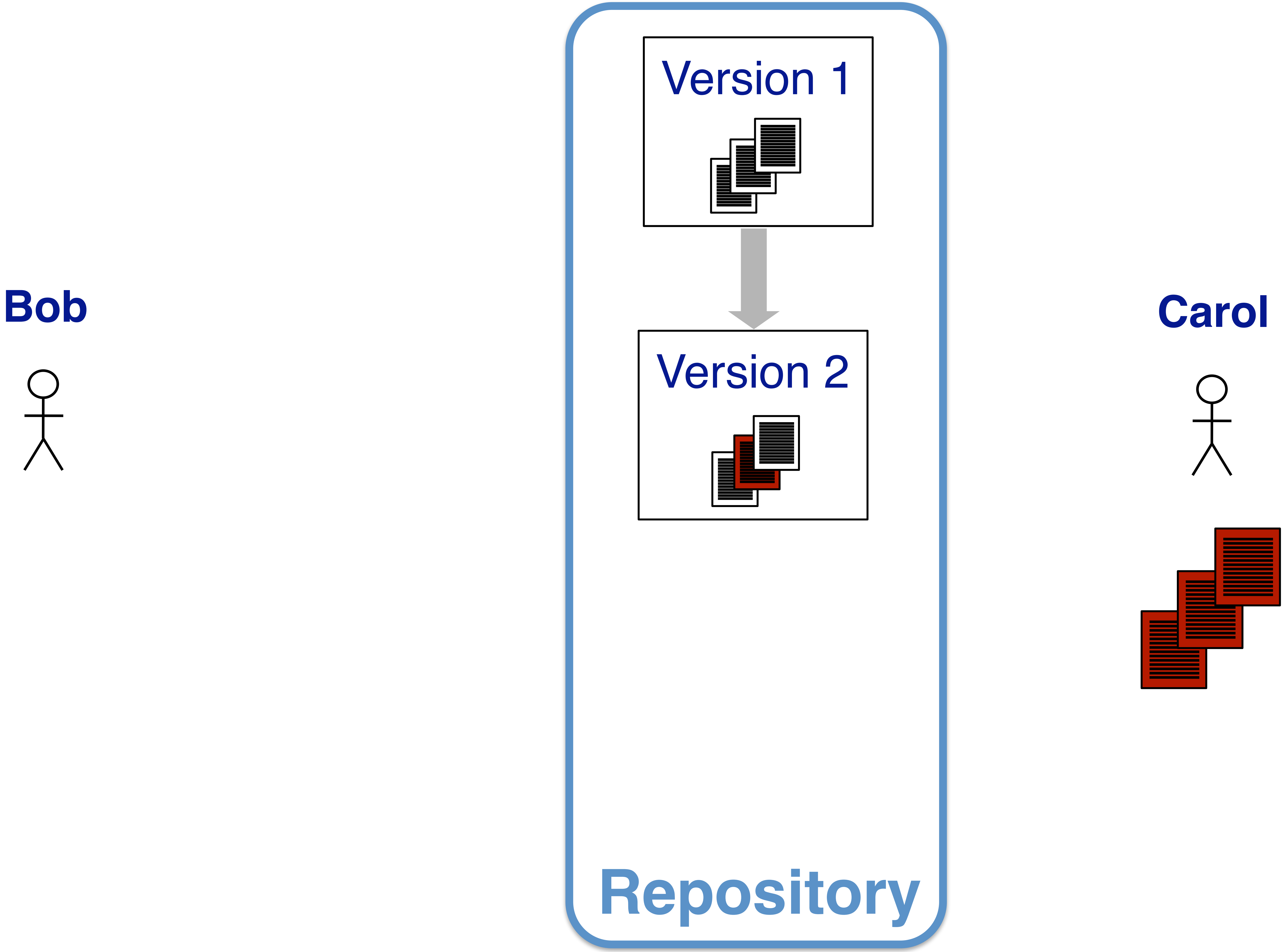
Bob

Version 1

Version 2

Repository

Carol



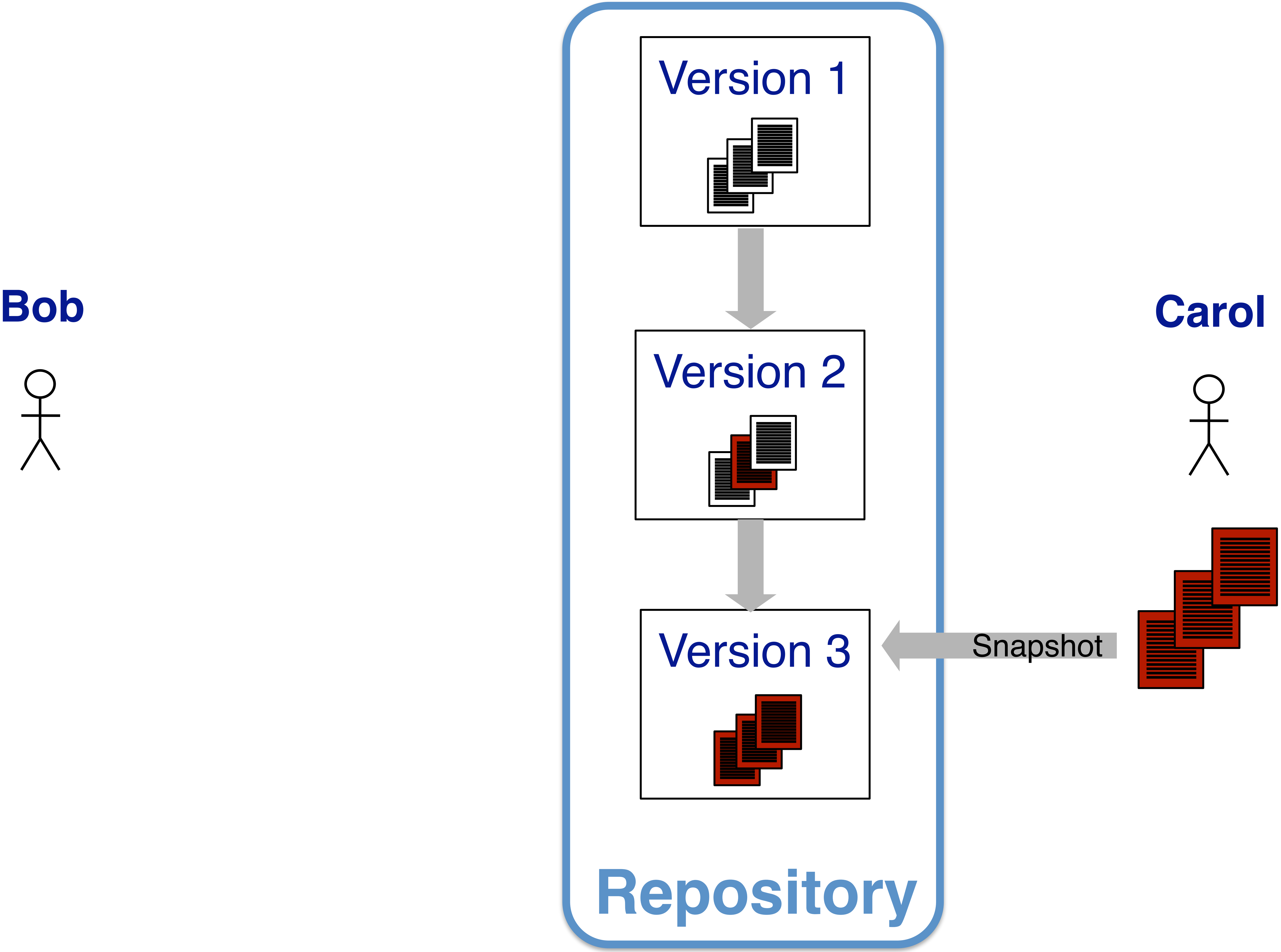
Bob

Version 1

Version 2

Repository

Carol



Bob

Version 1

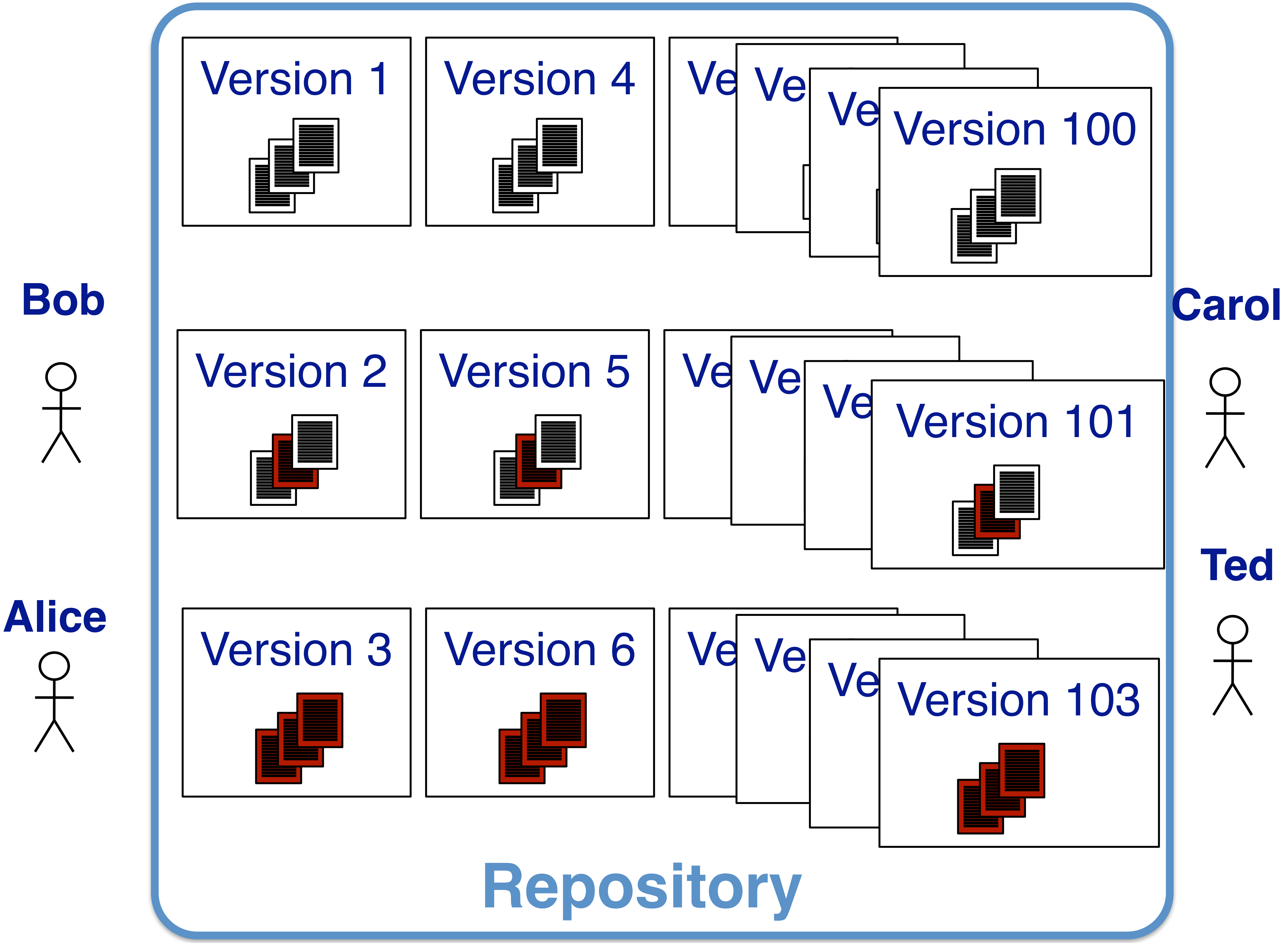
Version 2

Version 3

Snapshot

Repository

Carol


Version 1
Version 4
Ve
Ve
Ve
Version 100
Bob
Carol
Version 2
Version 5
Ve
Ve
Ve
Version 101
Ted
Alice
Version 3
Version 6
Ve
Ve
Ve
Version 103
Repository

Version 1
Version 4
Version 100
Bob
Carol
Version 2
Version 5
Version 101
Ted
Alice
Version 103
Repository

Controlled evolution

Can still lead to disaster!

git

git

Tracks the history of a collection of files

git

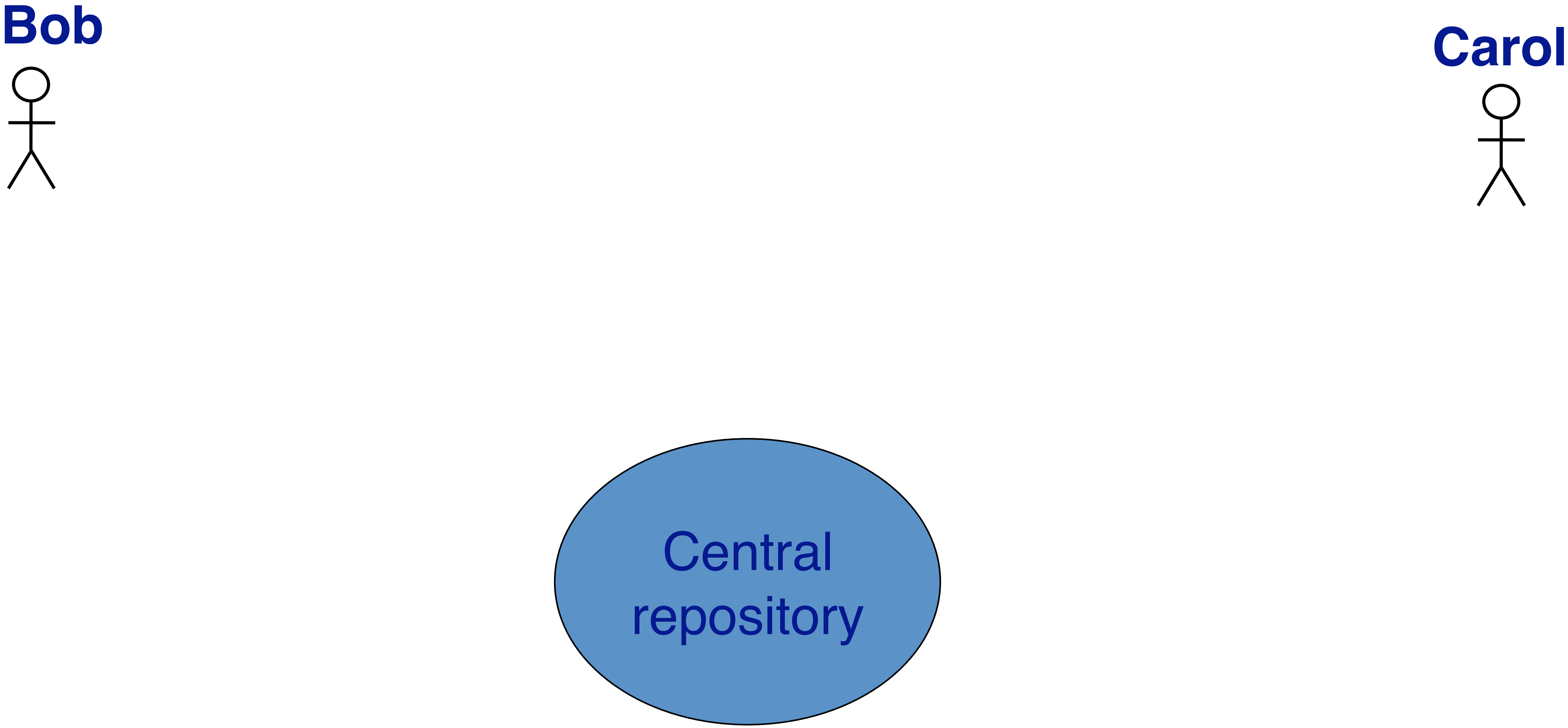
Tracks the history of a collection of files

Can revert the collection of files to another version

git

distributed
version control
system

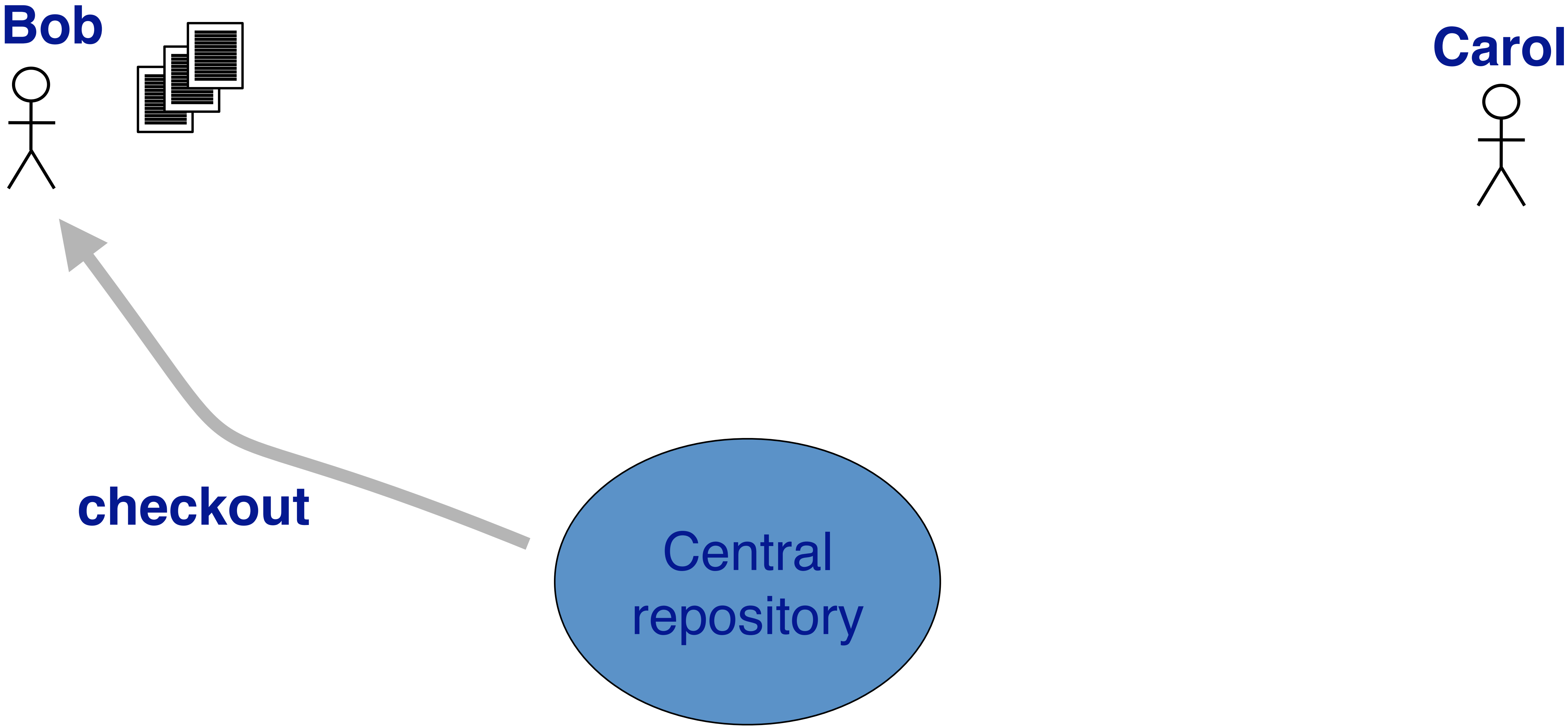
What is a centralized version control system?



Bob

Carol

Central repository

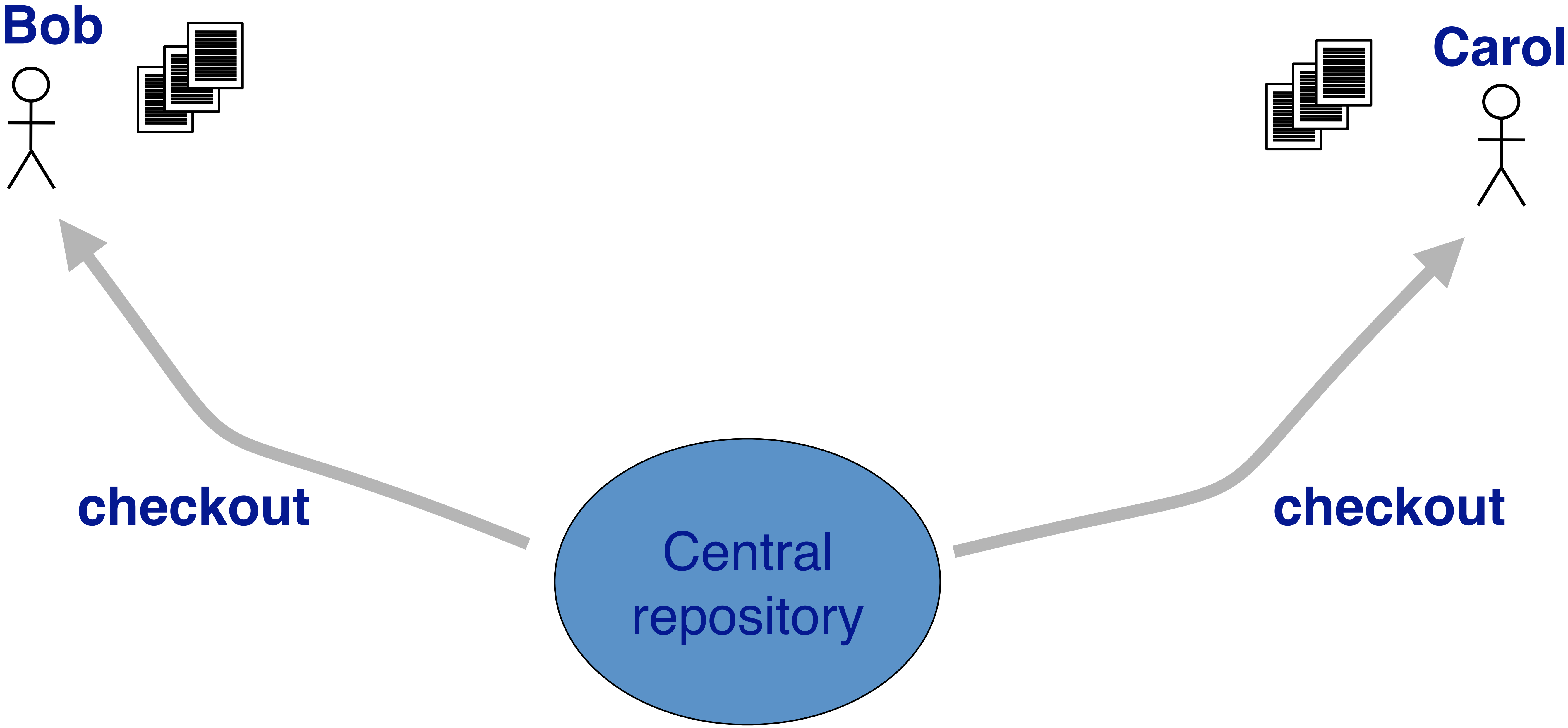


Bob

Carol

checkout

Central repository



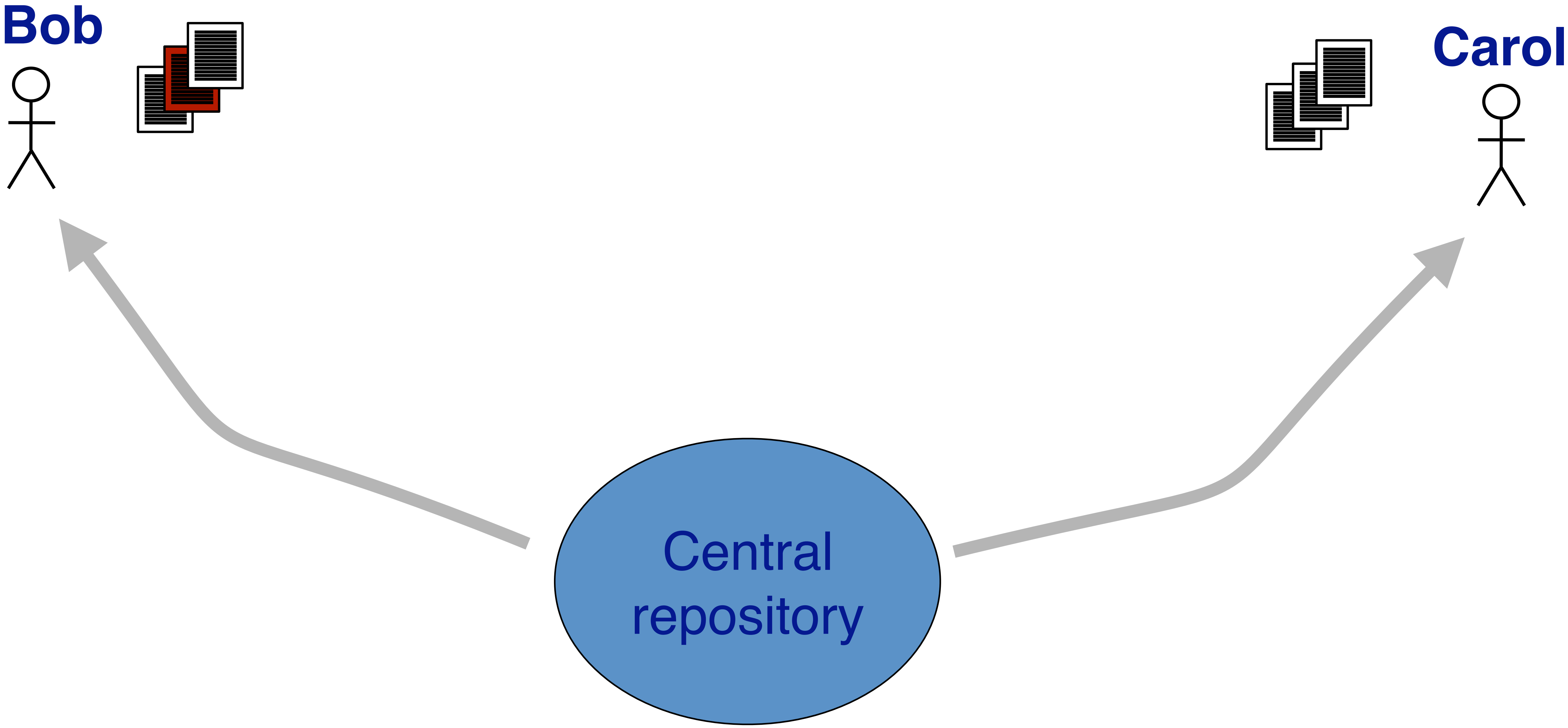
Bob

Carol

checkout

checkout

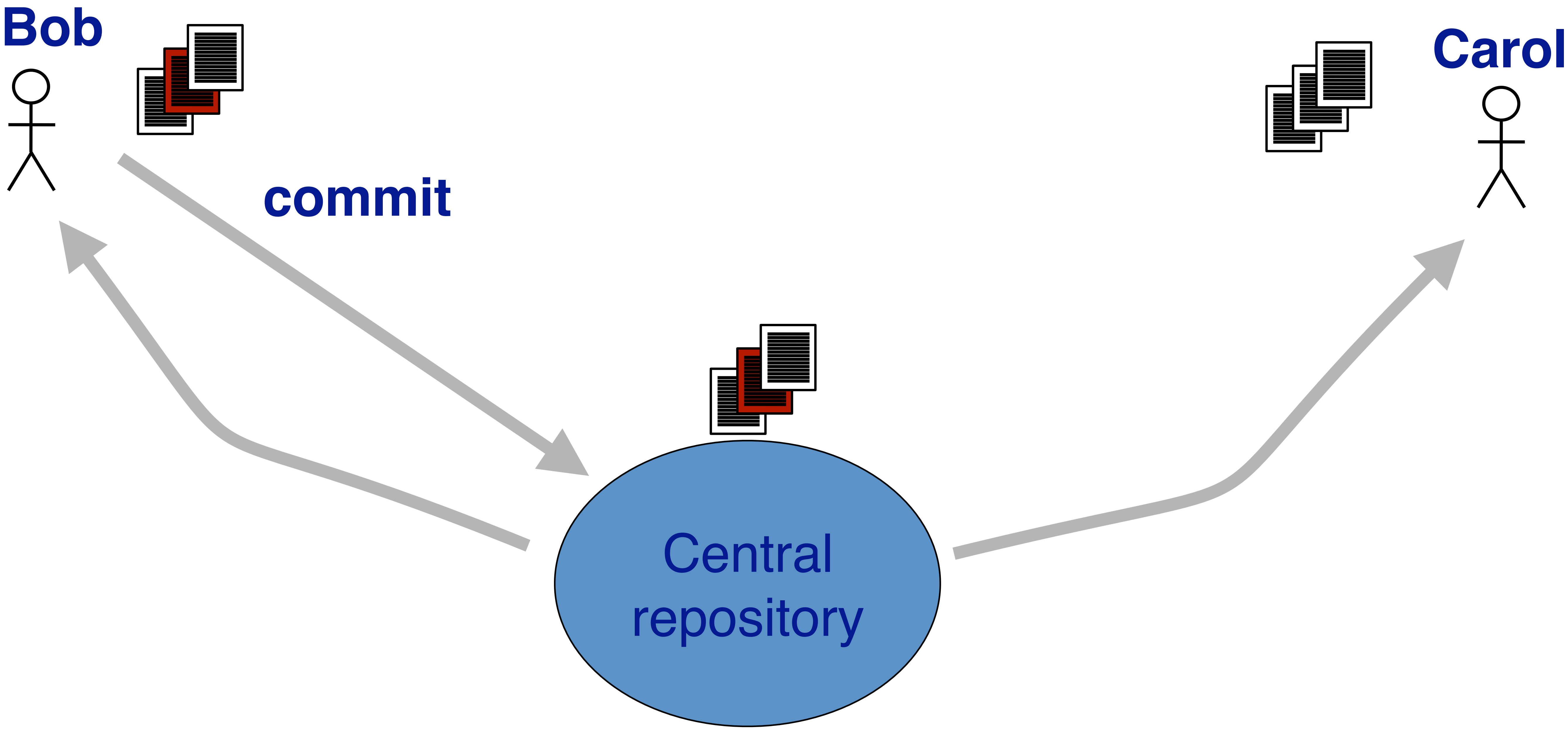
Central repository

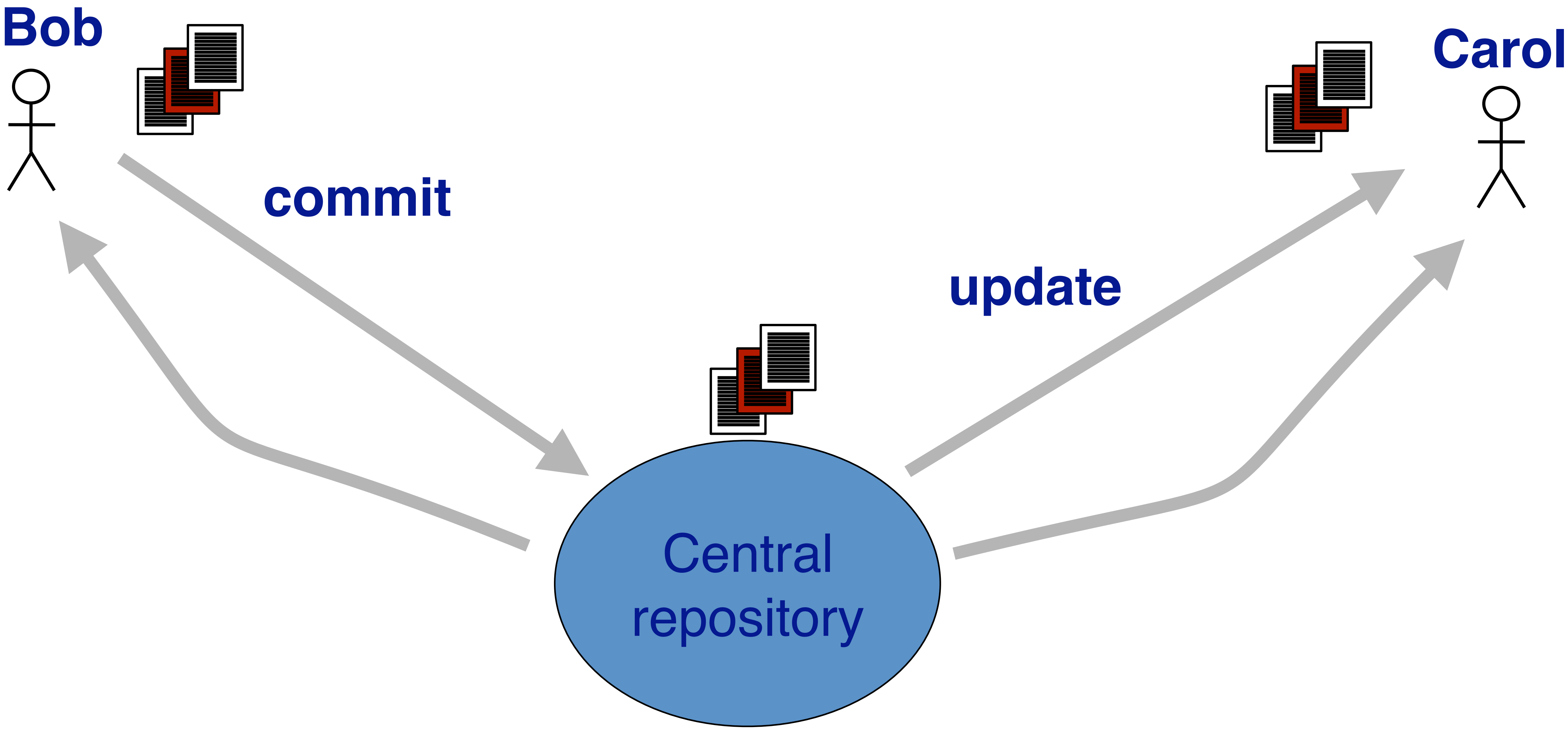


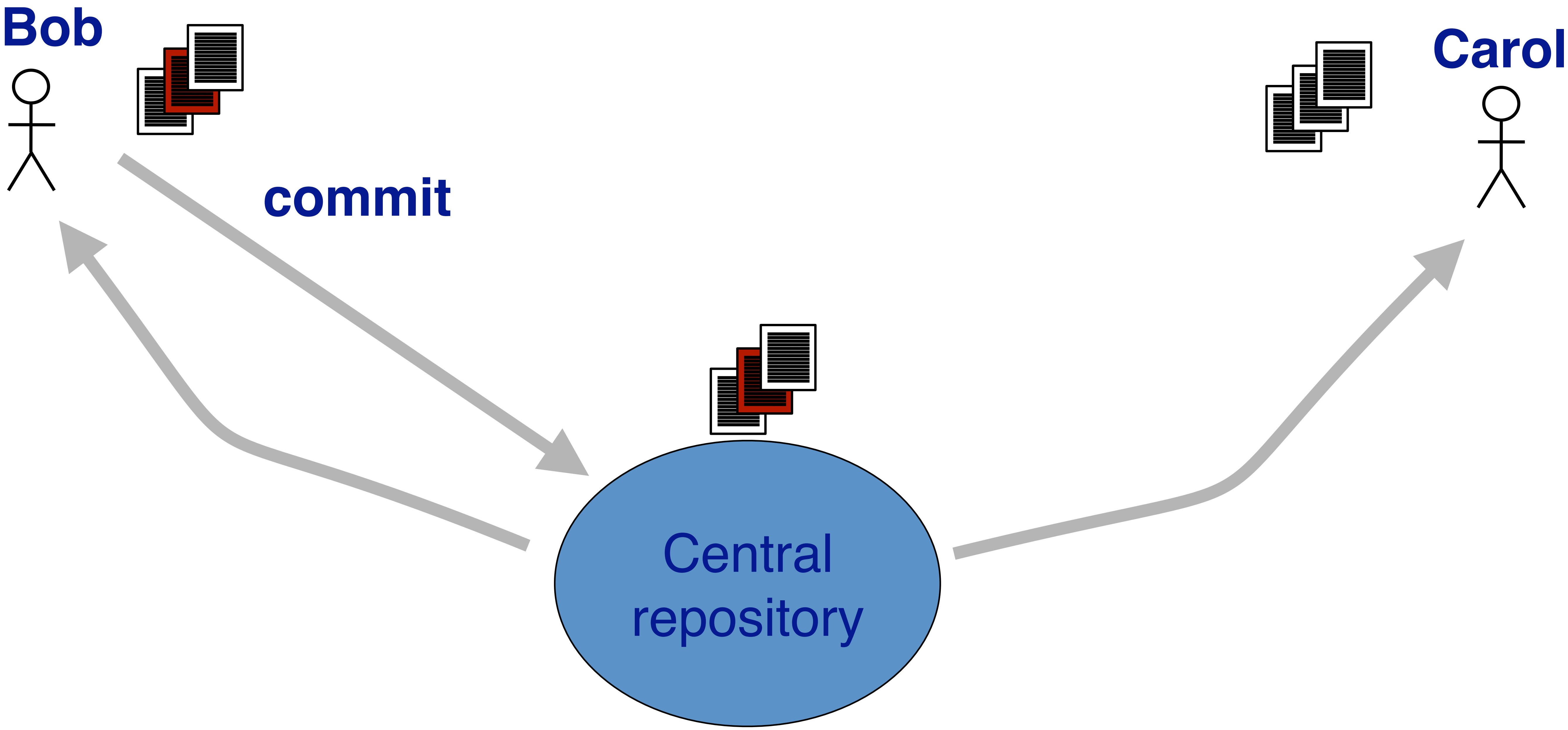
Bob

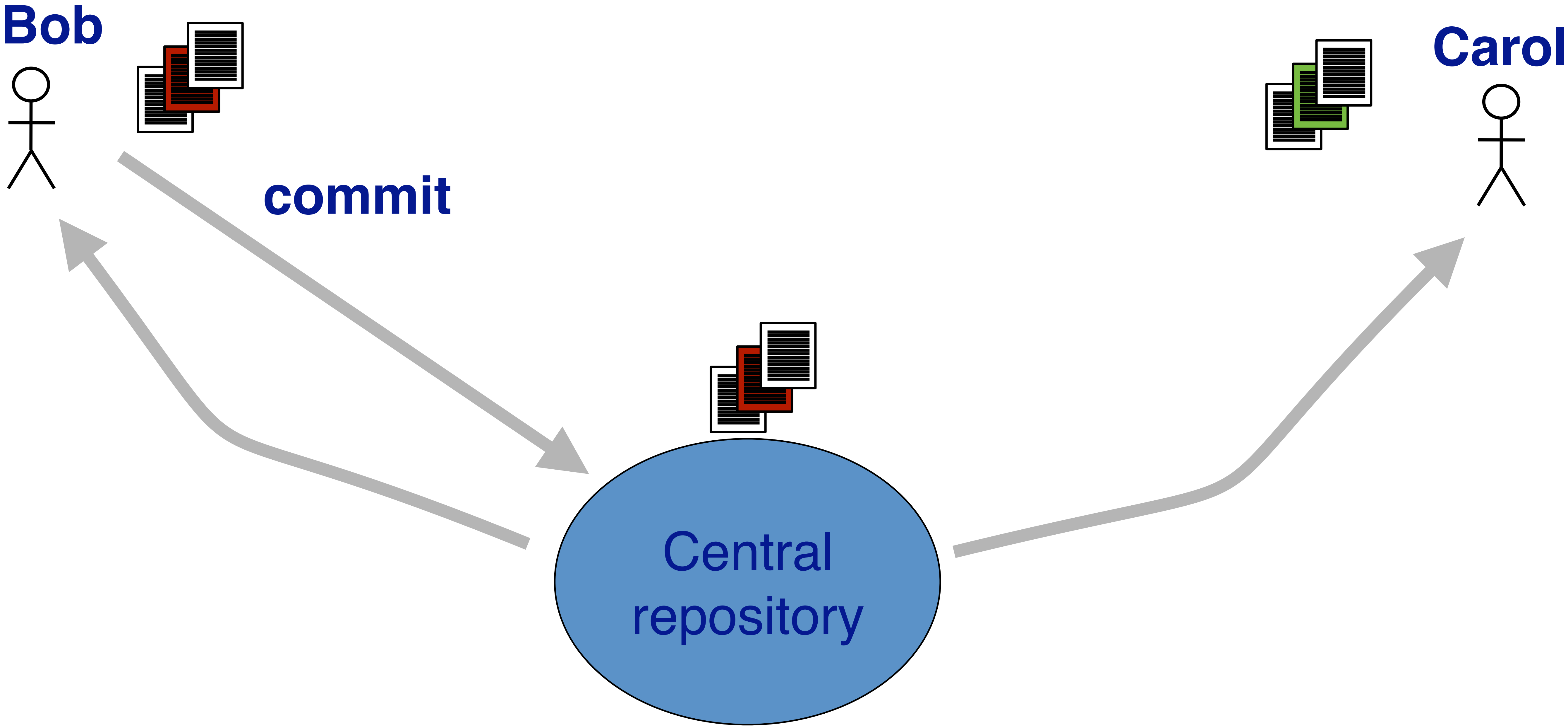
Carol

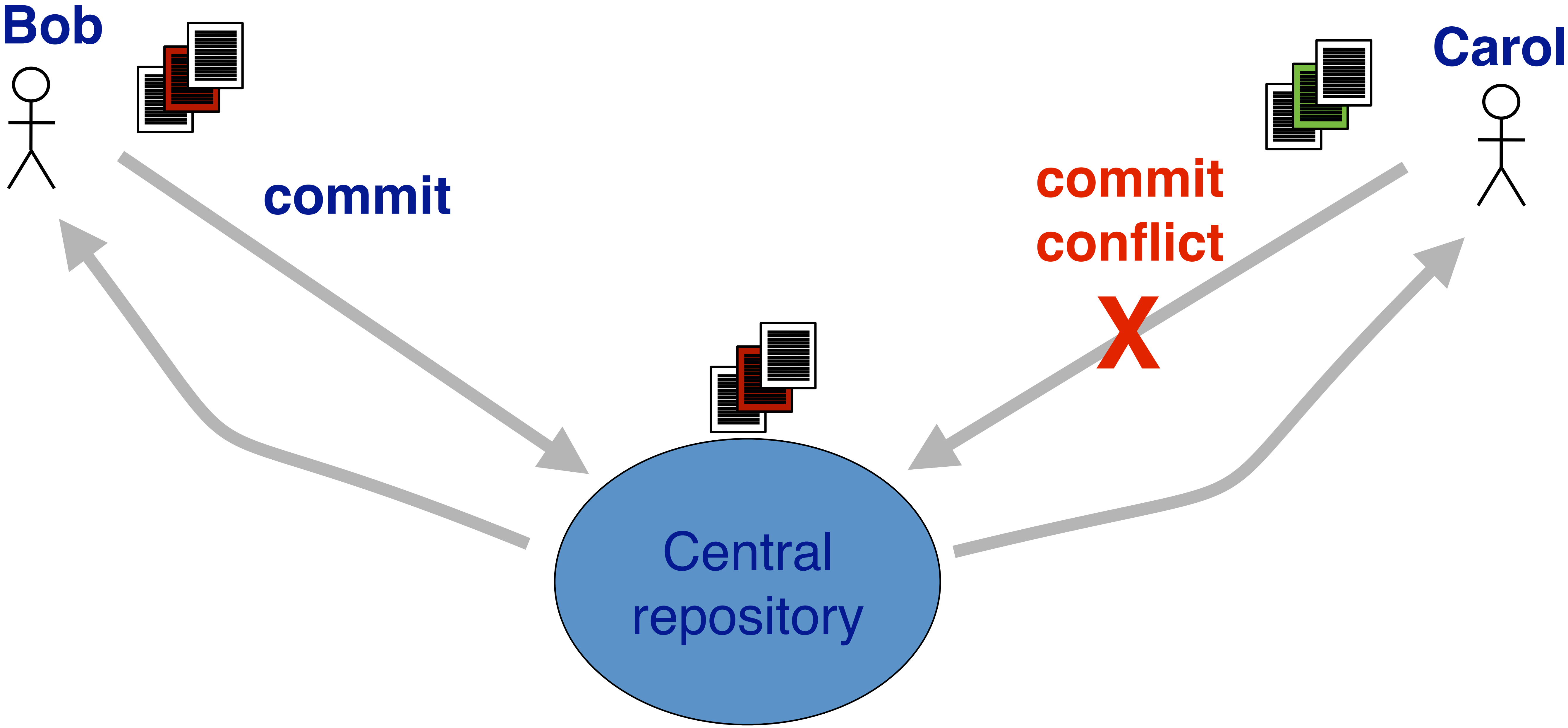
Central repository

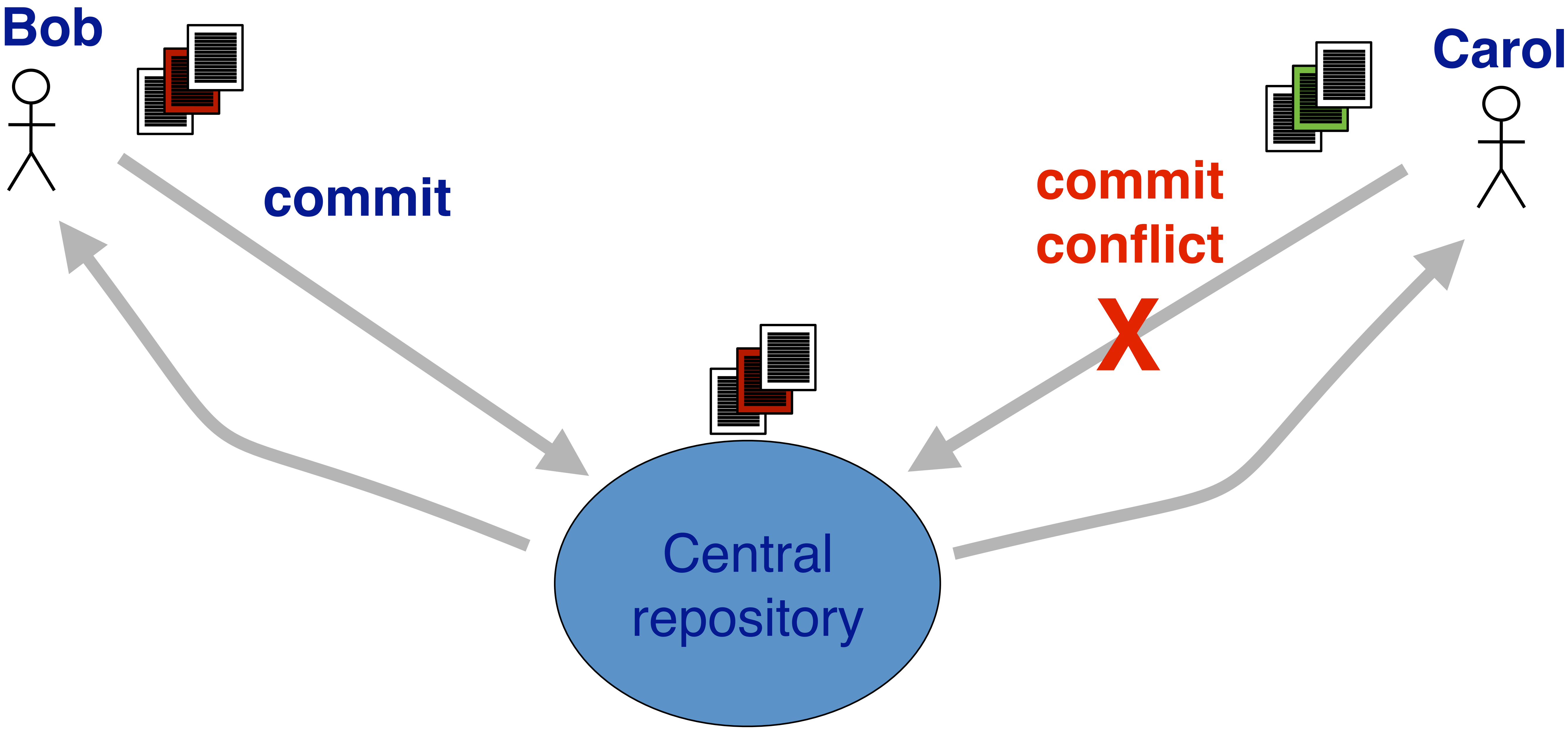

Bob
commit
Carol
Central repository


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

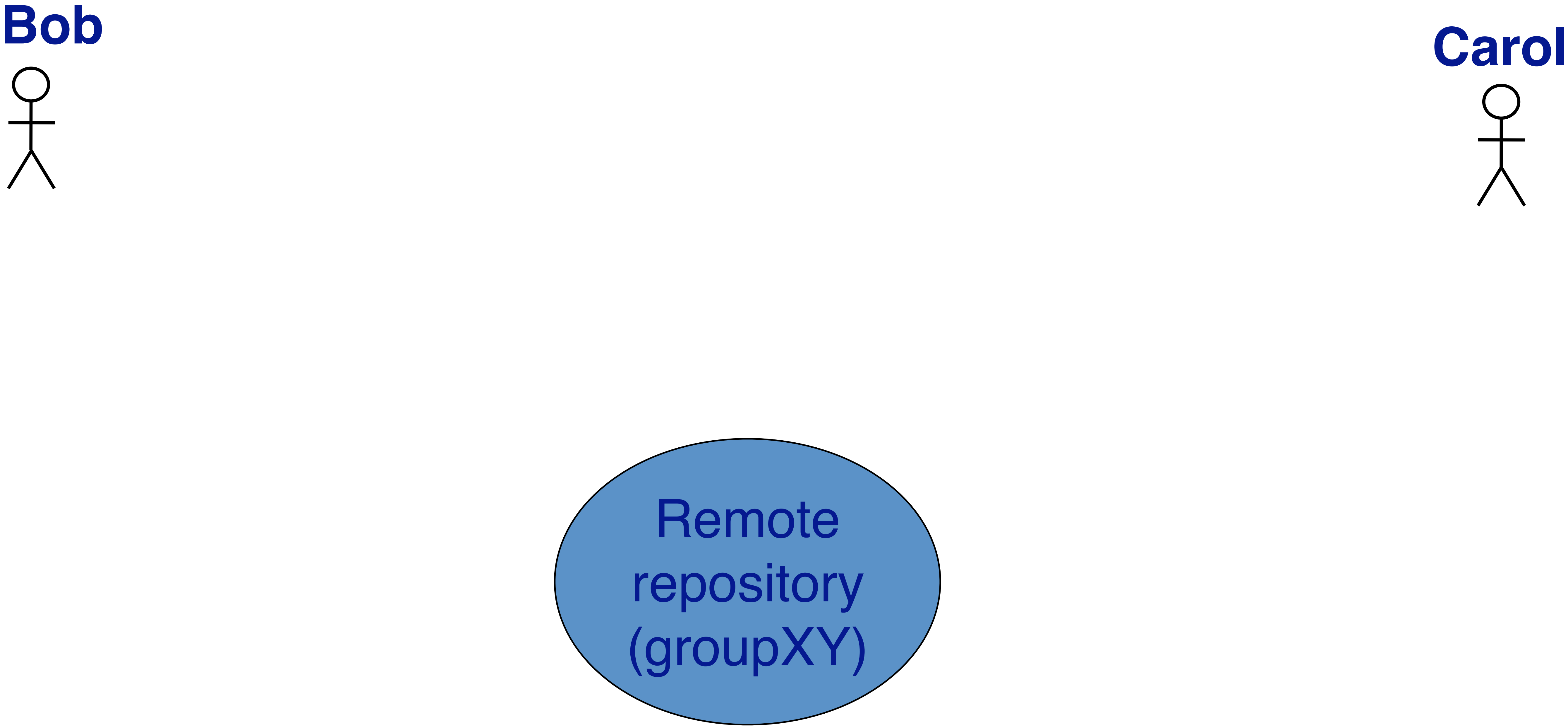
conflict

X

Central repository

you must update before every commit

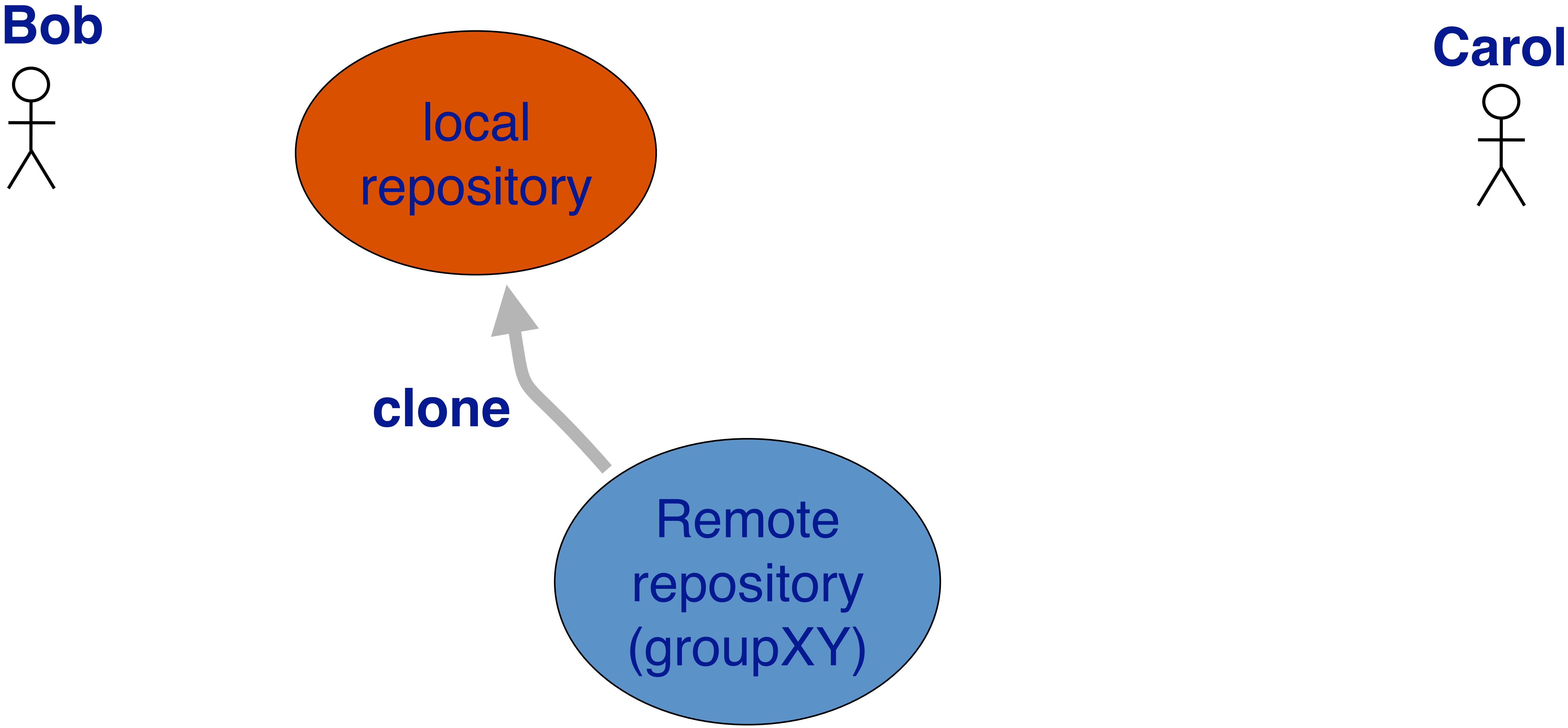
What is a distributed version control system?



Bob

Carol

Remote repository (groupXY)



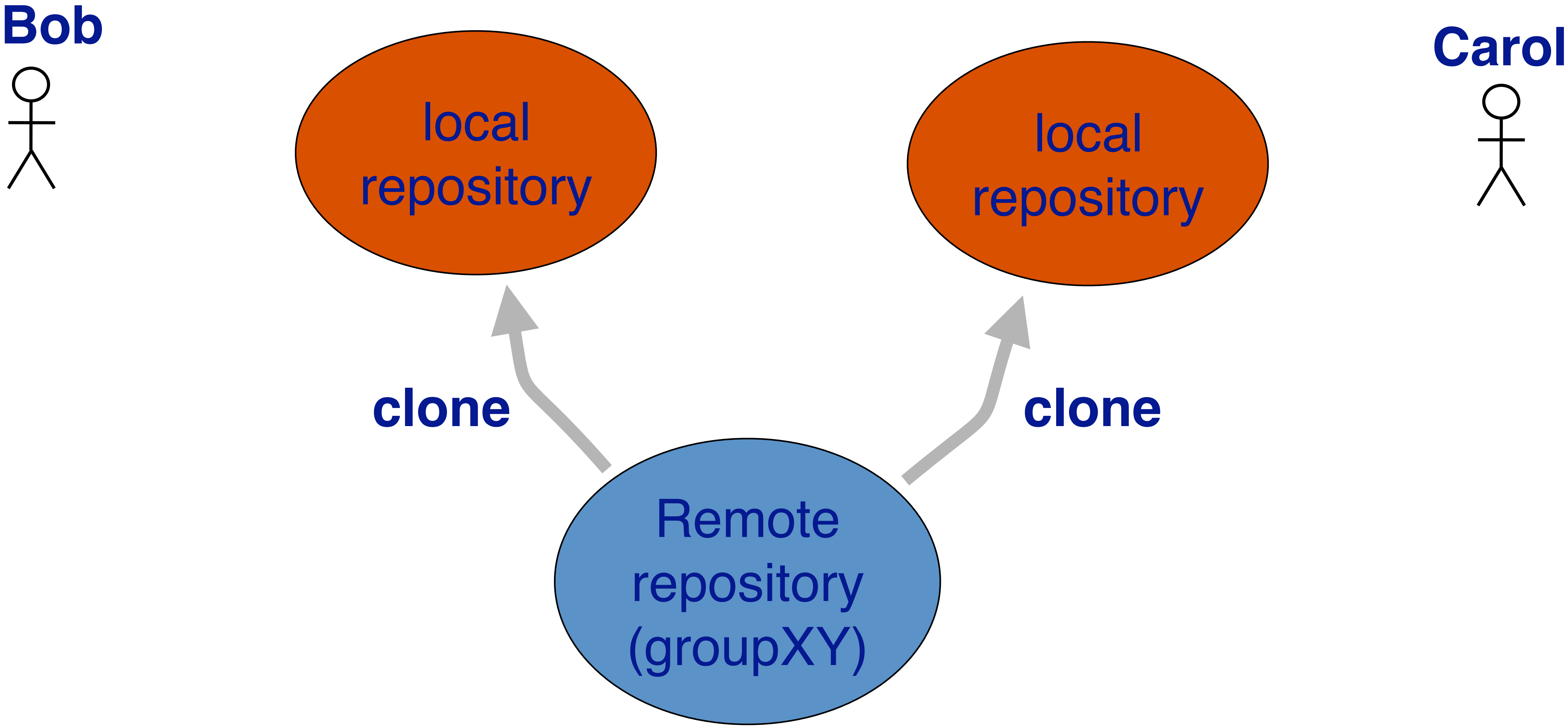
Bob

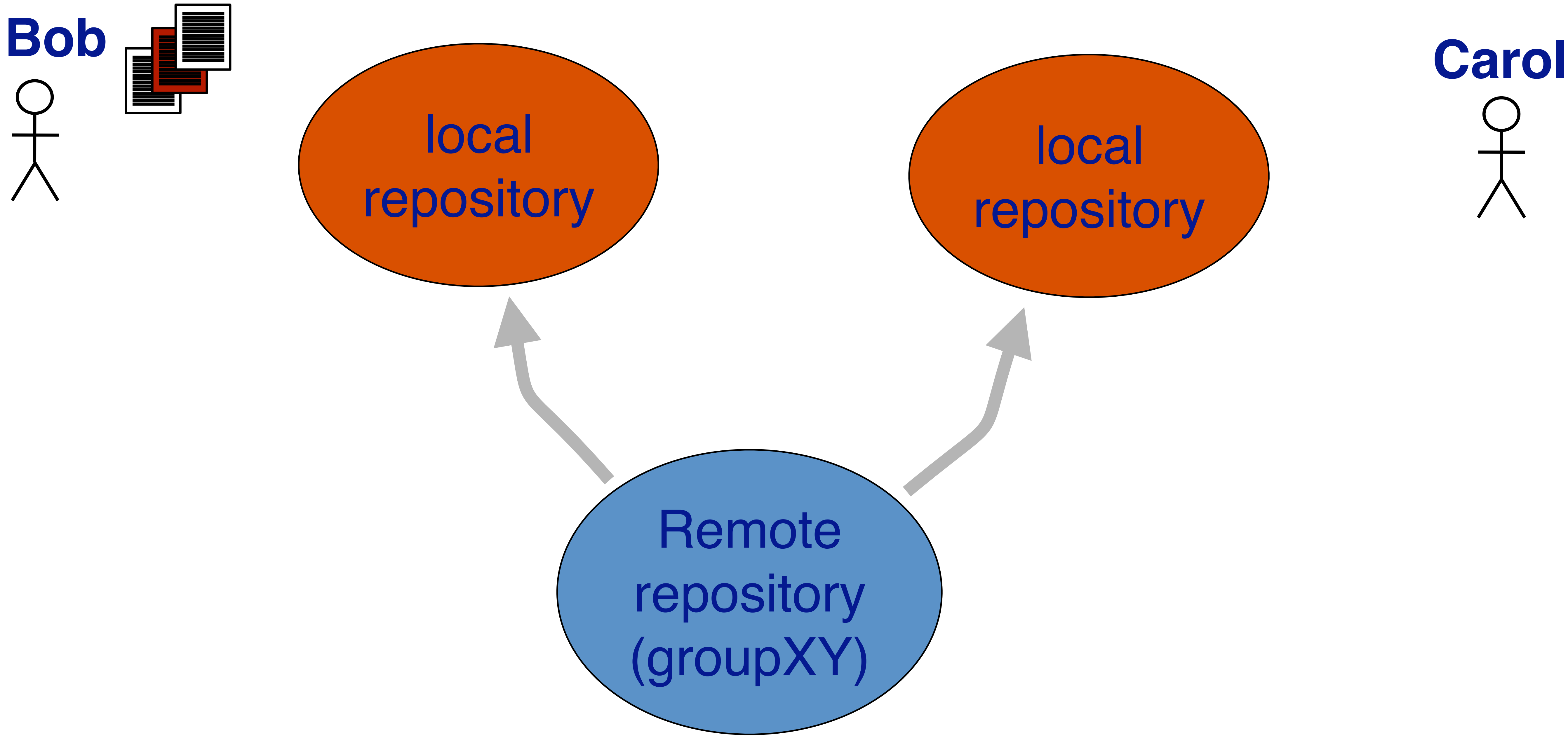
local repository

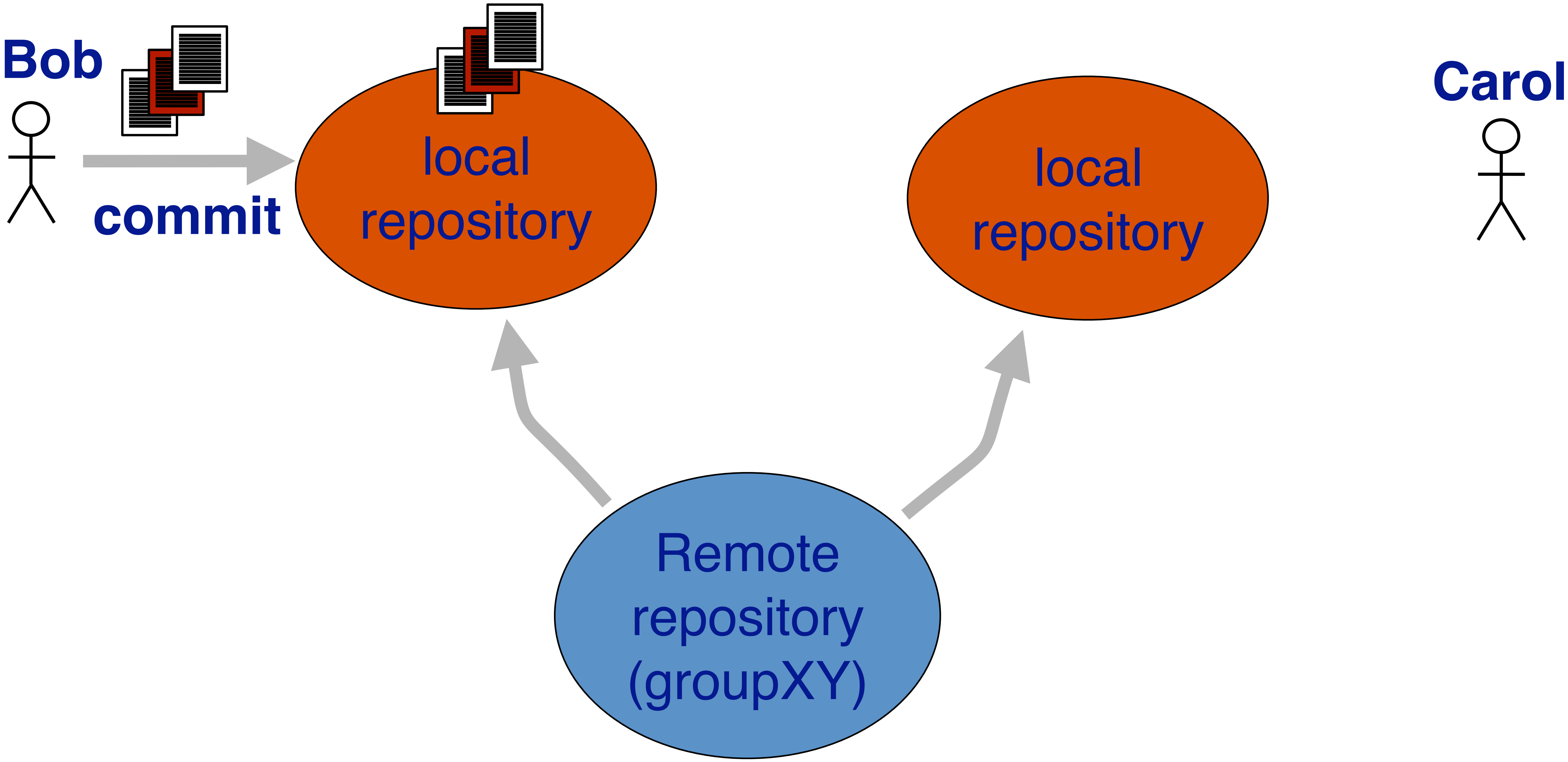
clone

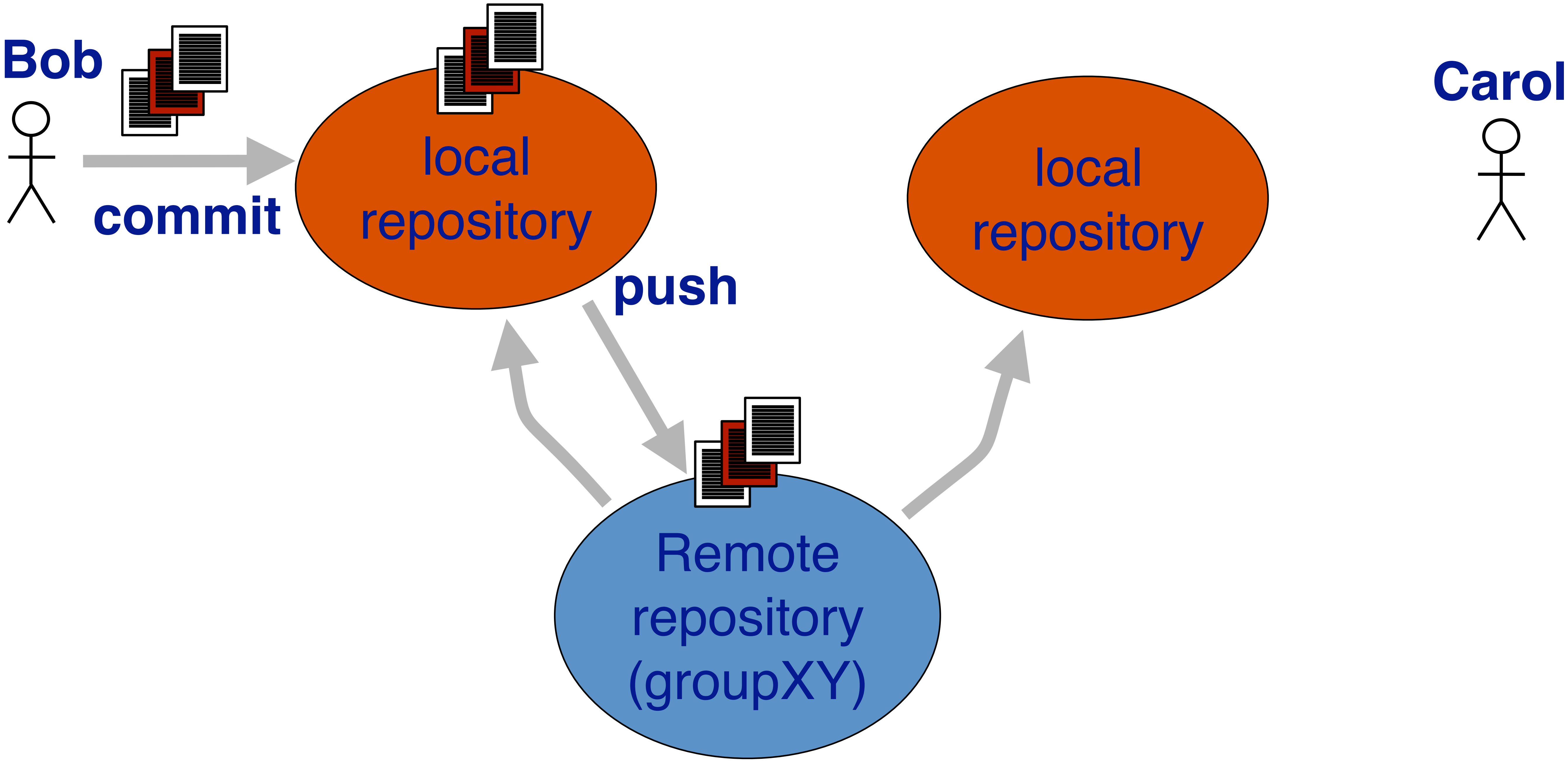
Remote repository (groupXY)

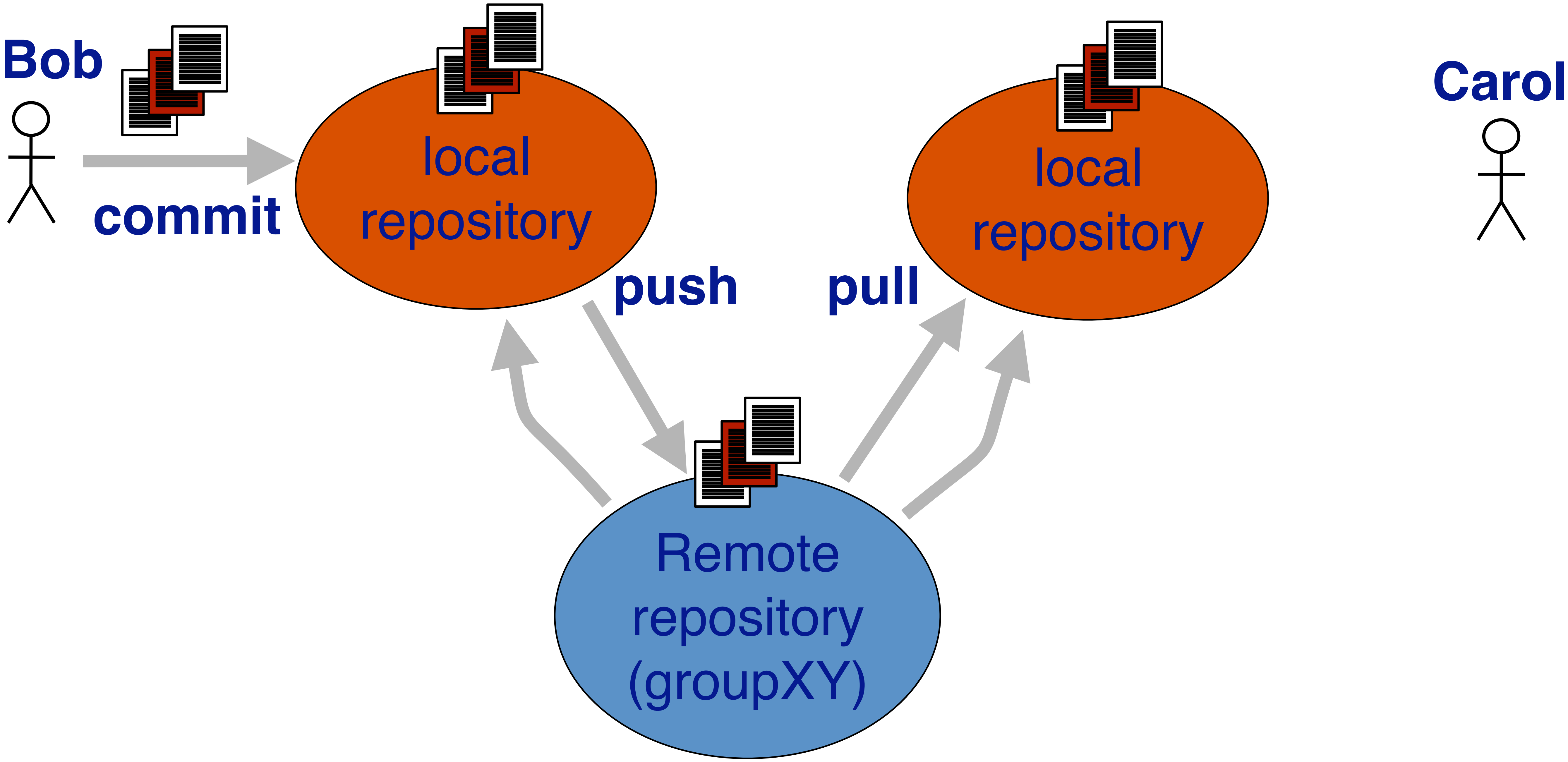
Carol


Bob
local repository
local repository
Carol
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

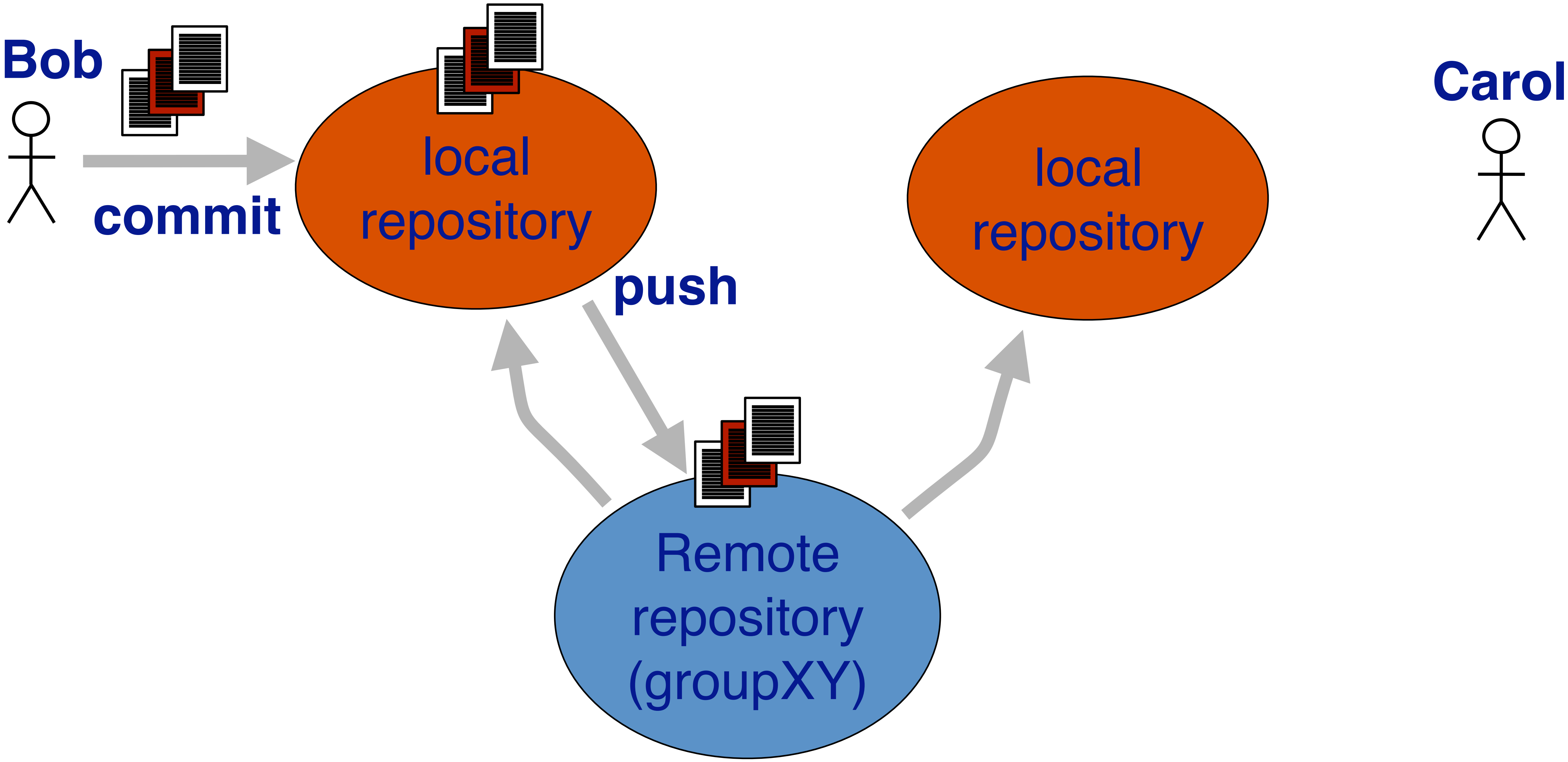
push

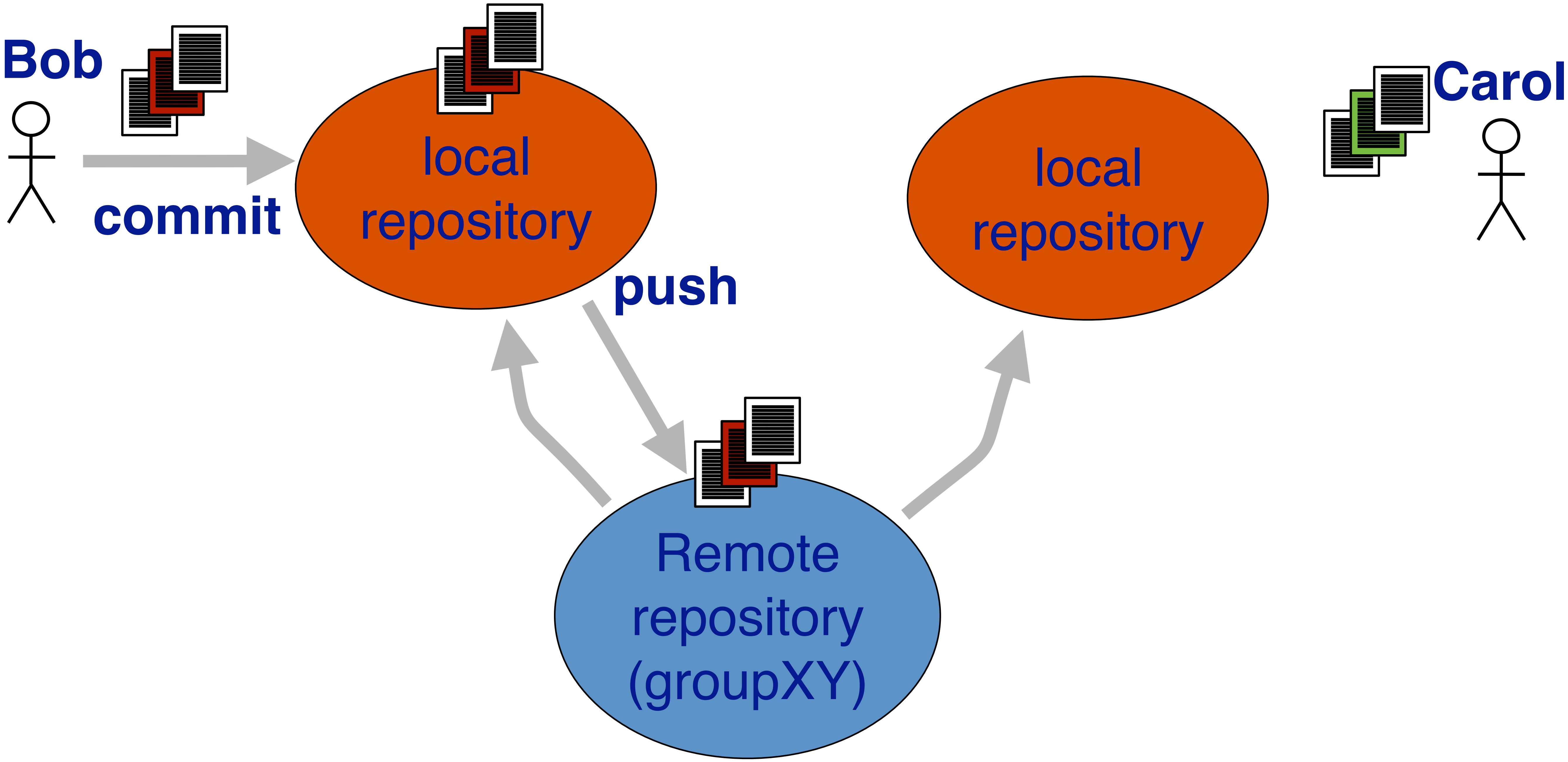
pull

local repository

Carol

Remote repository (groupXY)


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



Bob

commit

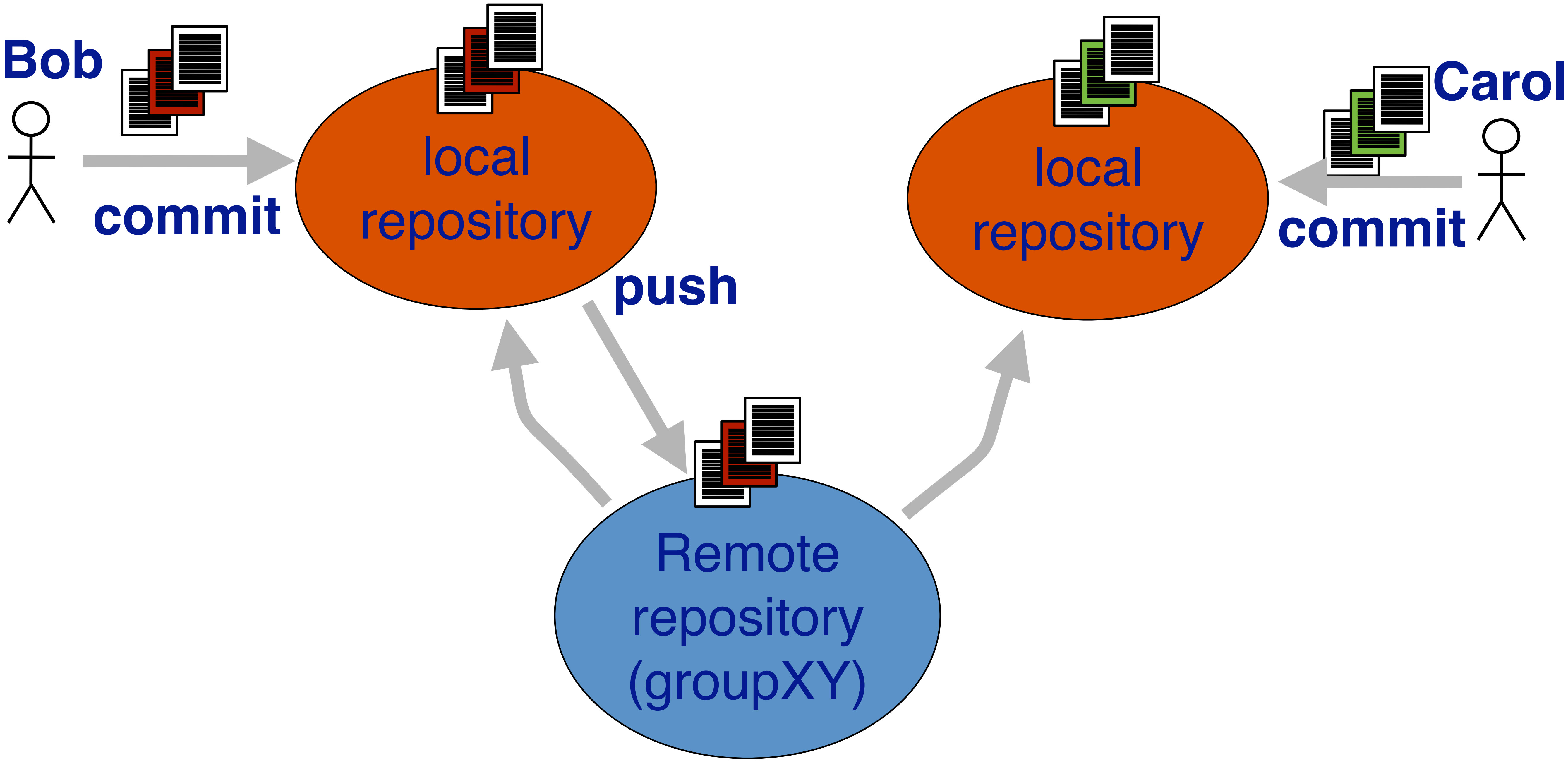
local repository

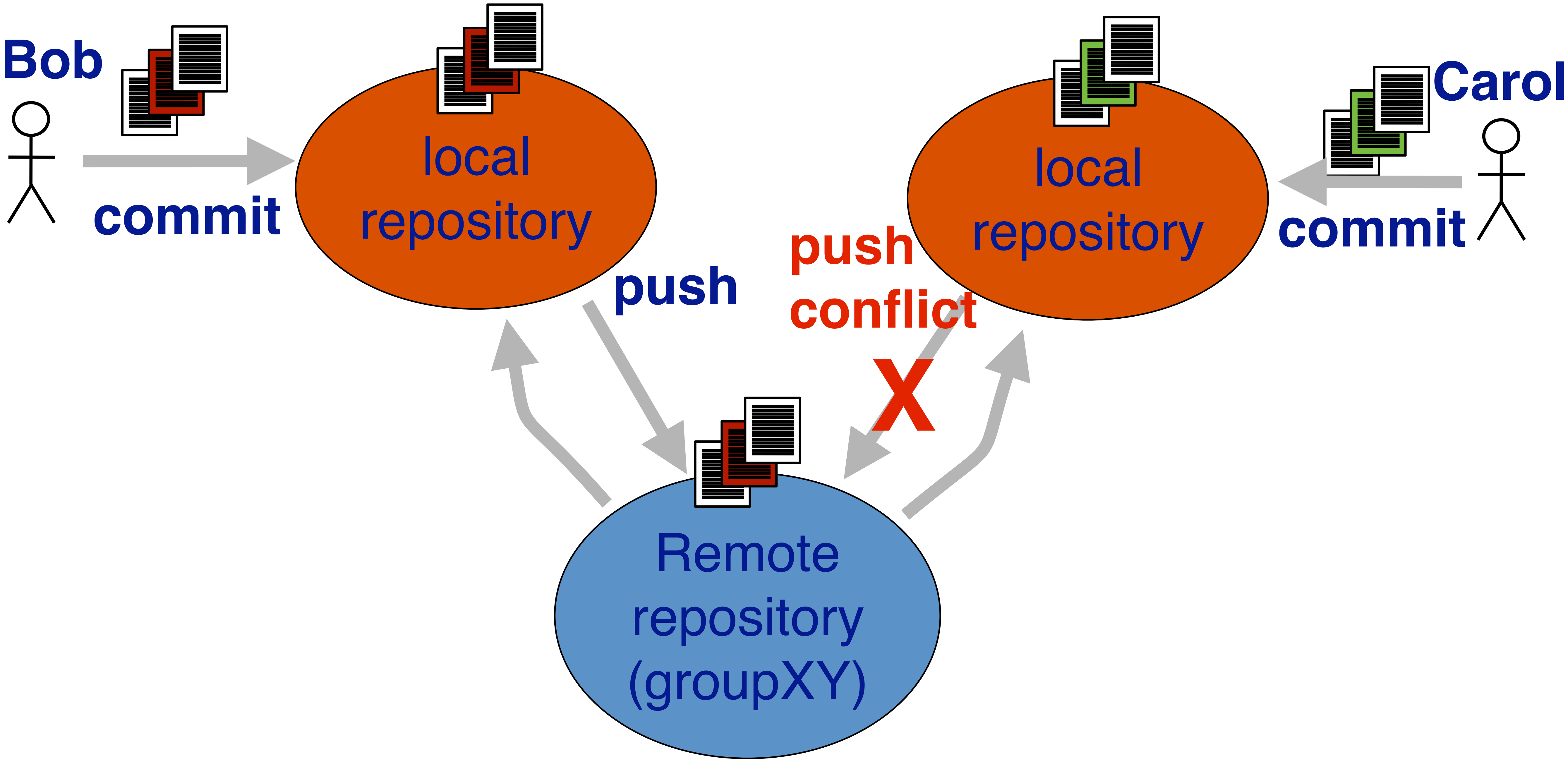
push

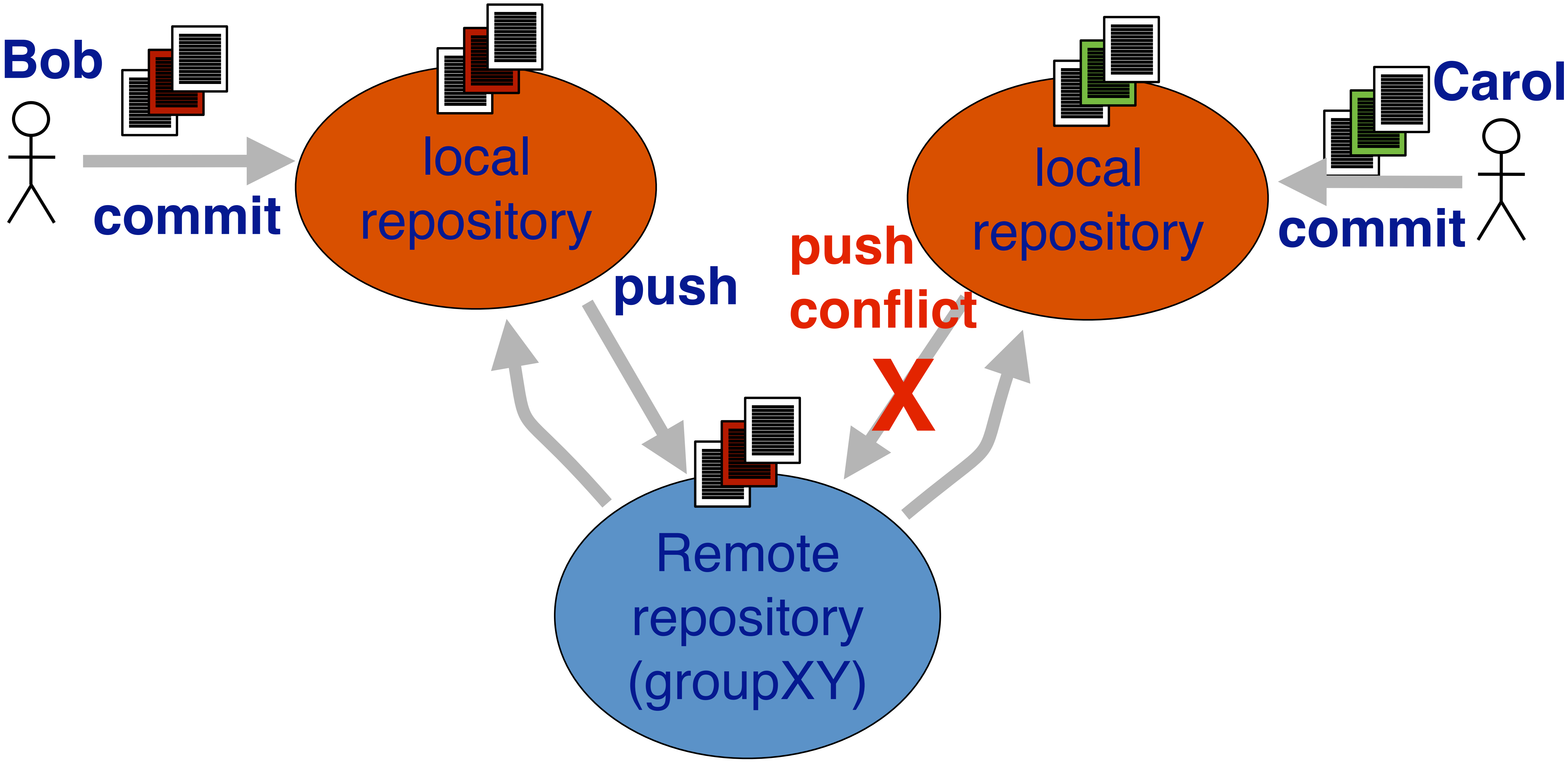
local repository

Carol

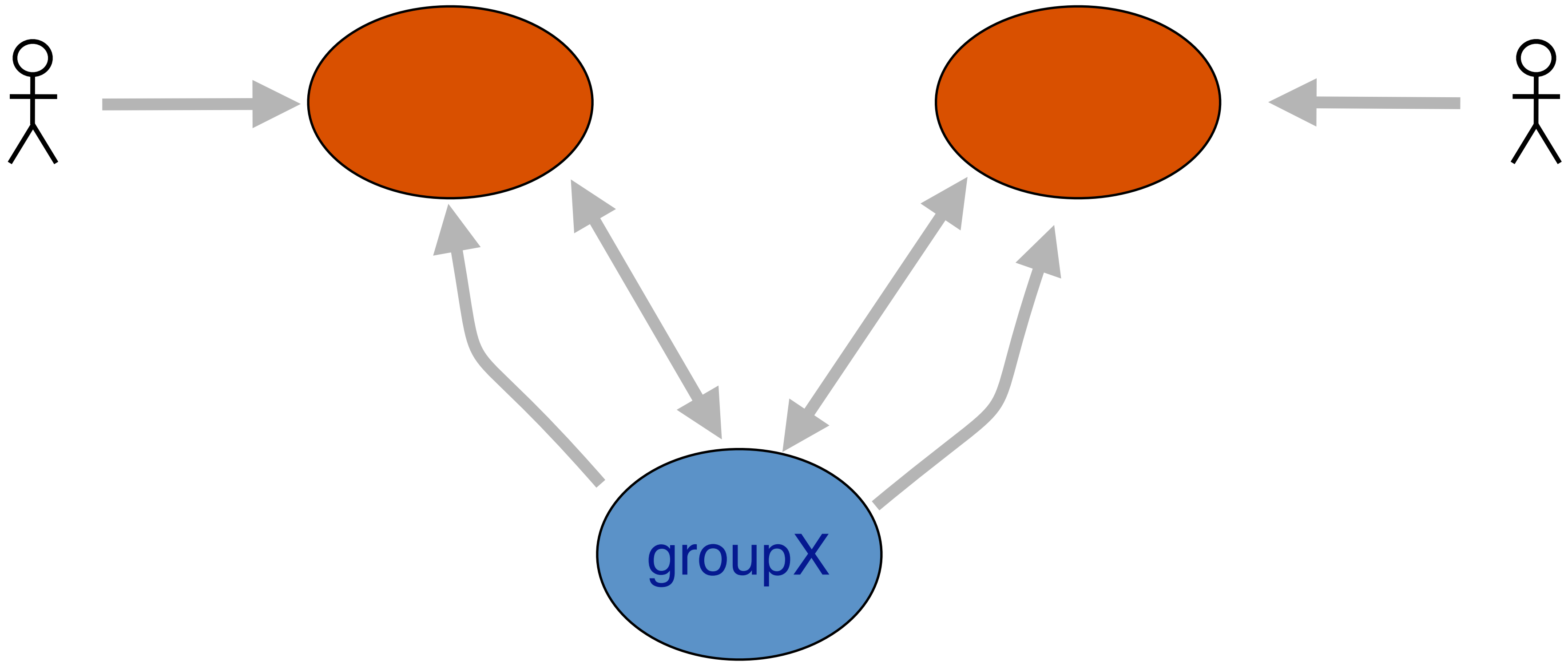
Remote repository (groupXY)

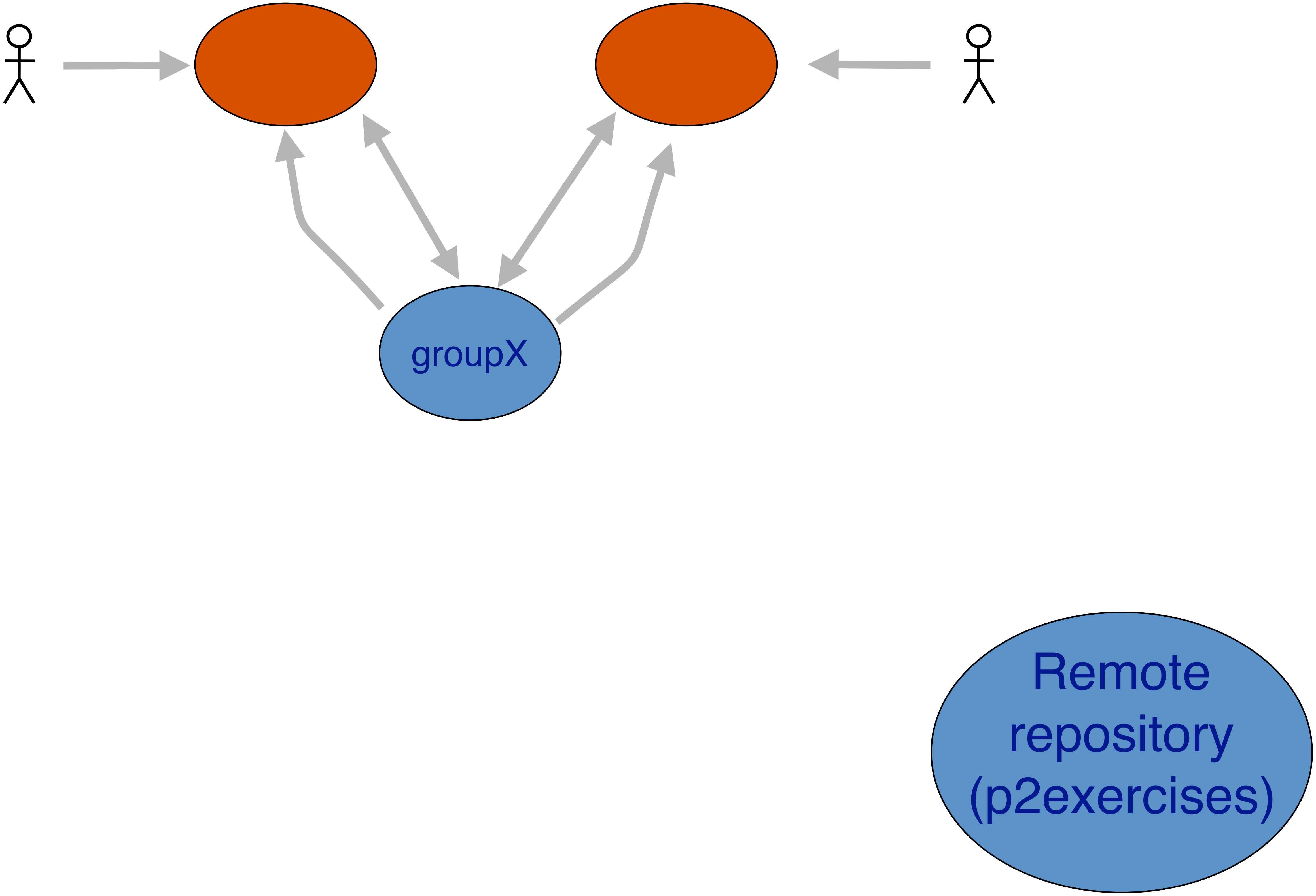

Bob
commit
local repository
push
local repository
Carol
commit
Remote repository (groupXY)


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



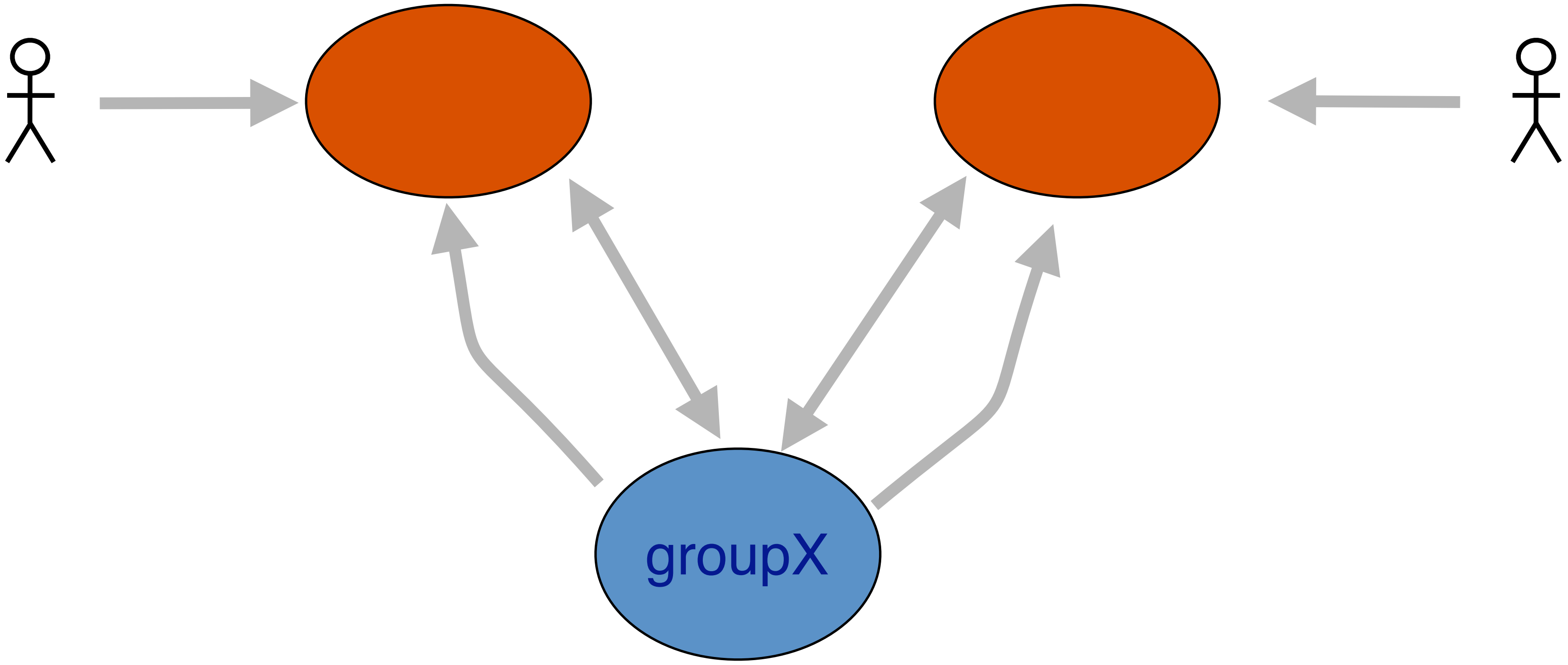
you must pull before every push

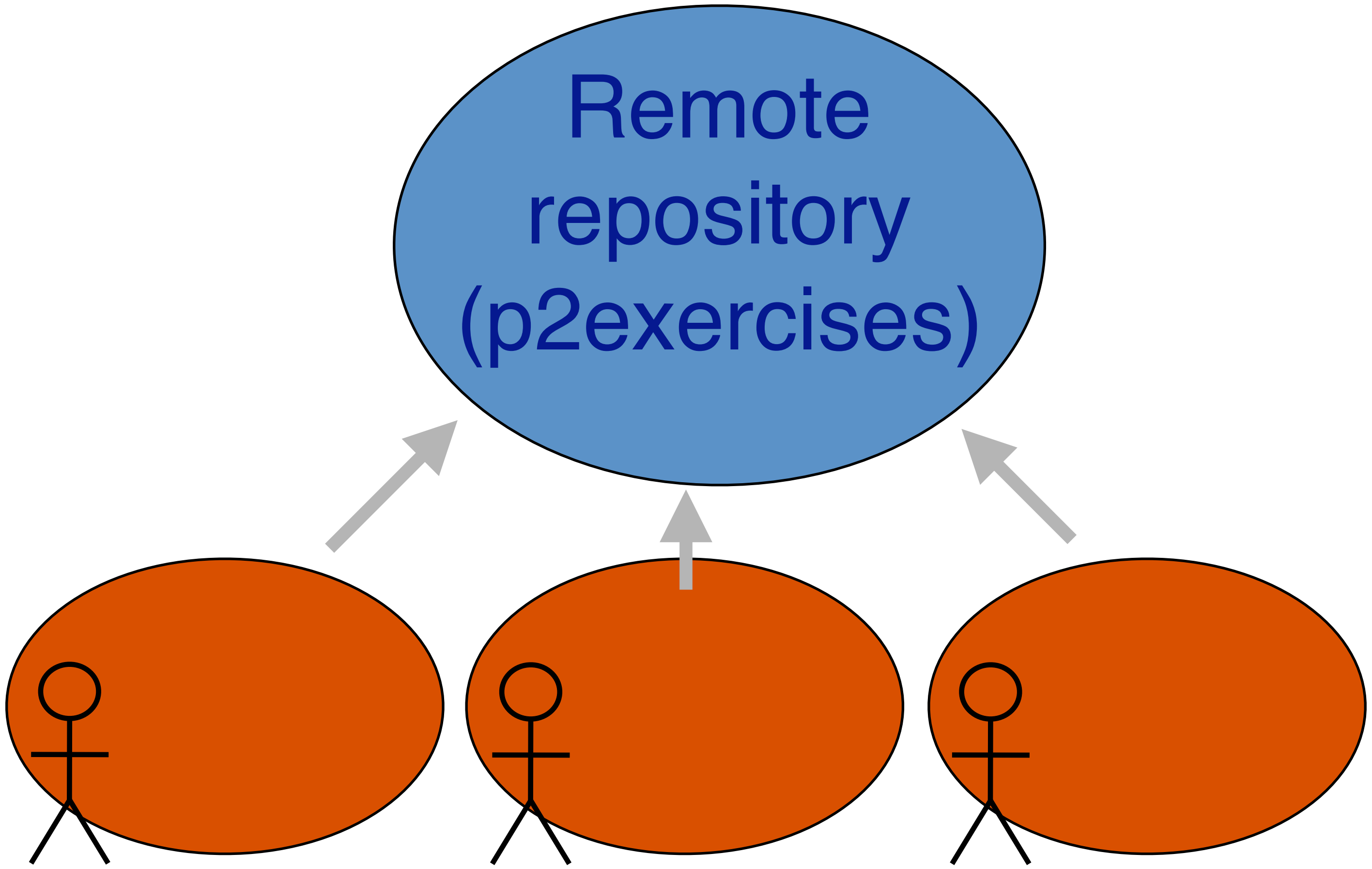

groupX

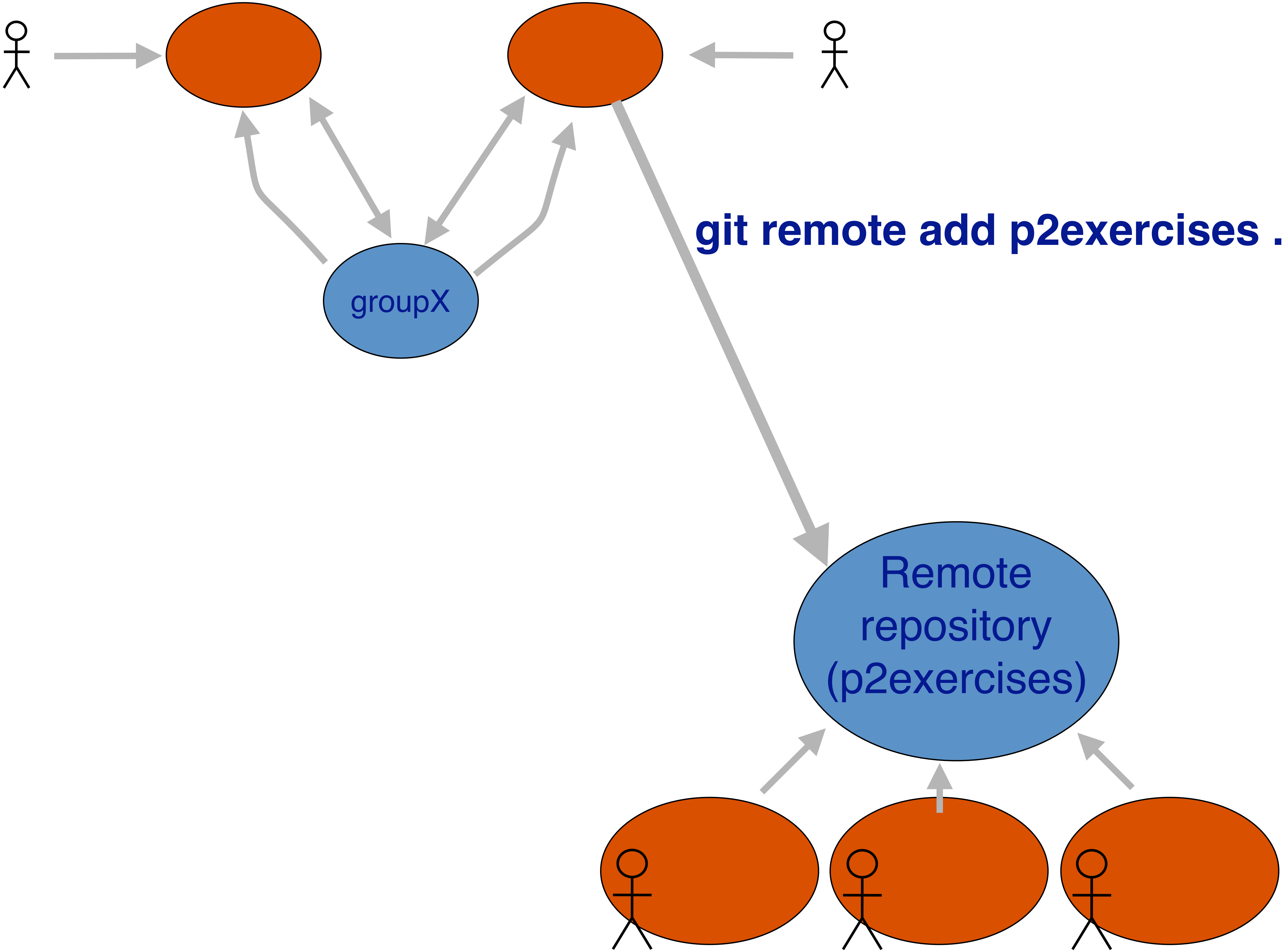


groupX

Remote repository (p2exercises)


groupX

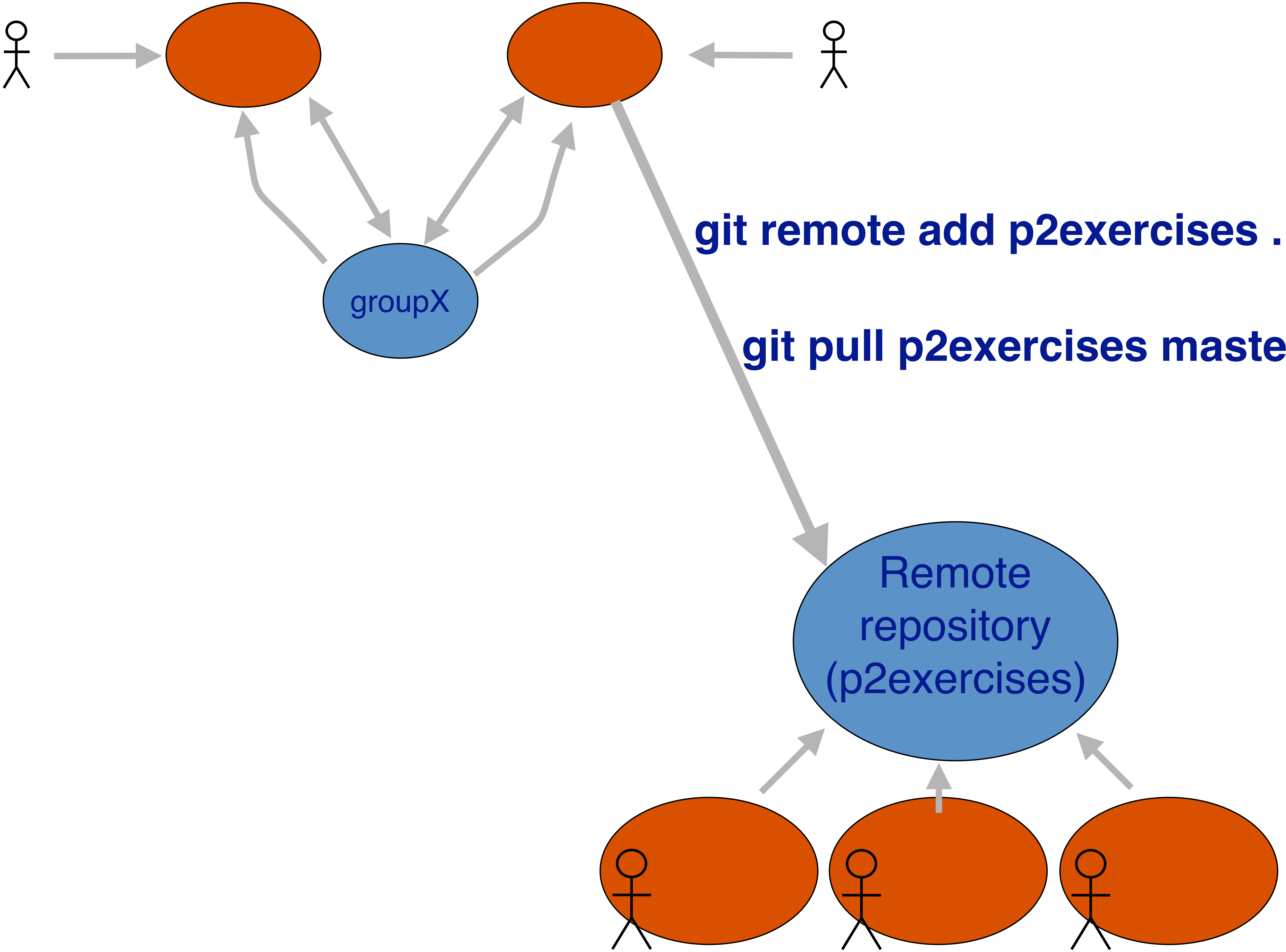

Remote repository (p2exercises)



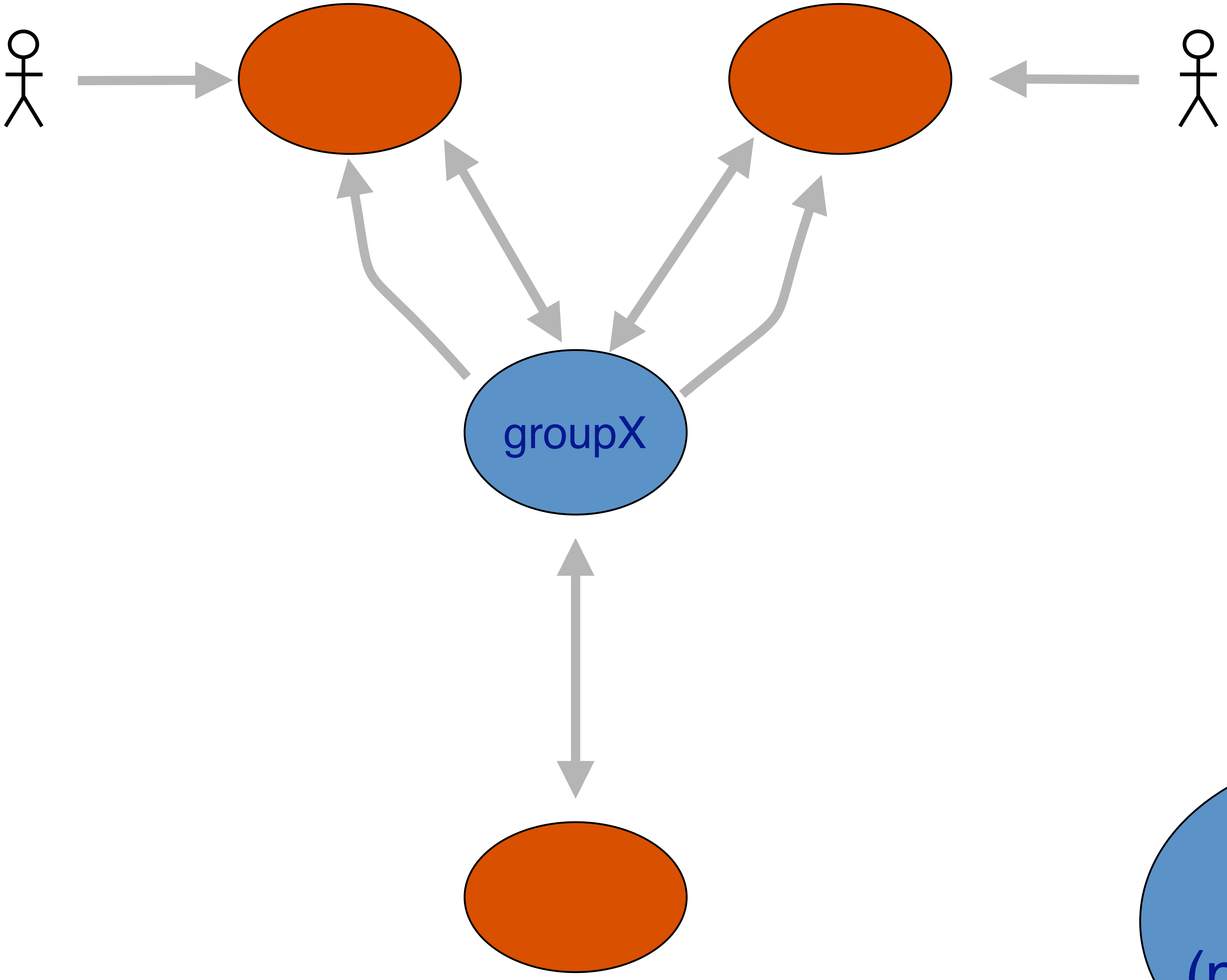
```
git remote add p2exercises ...
```

groupX

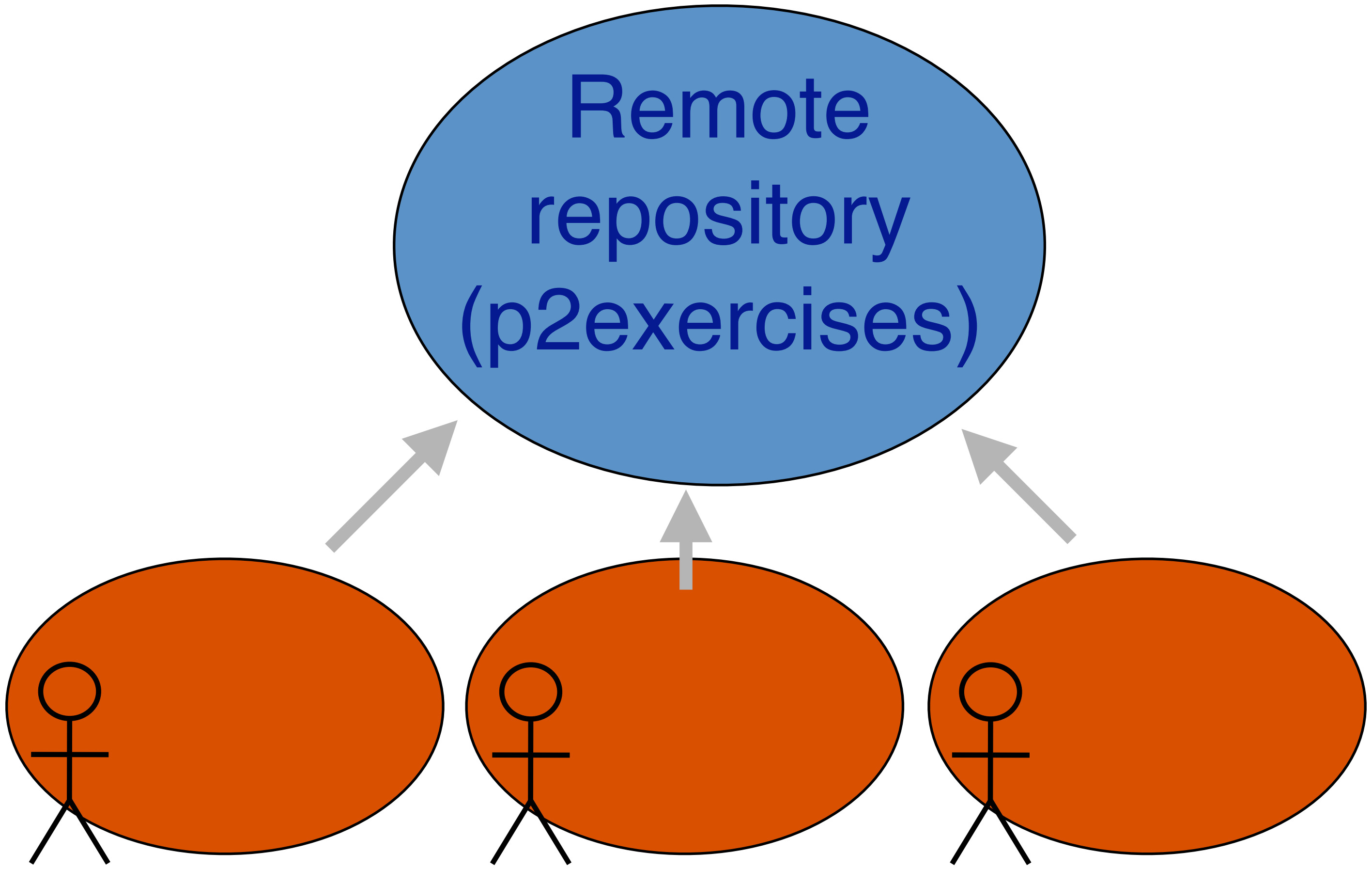
Remote repository (p2exercises)



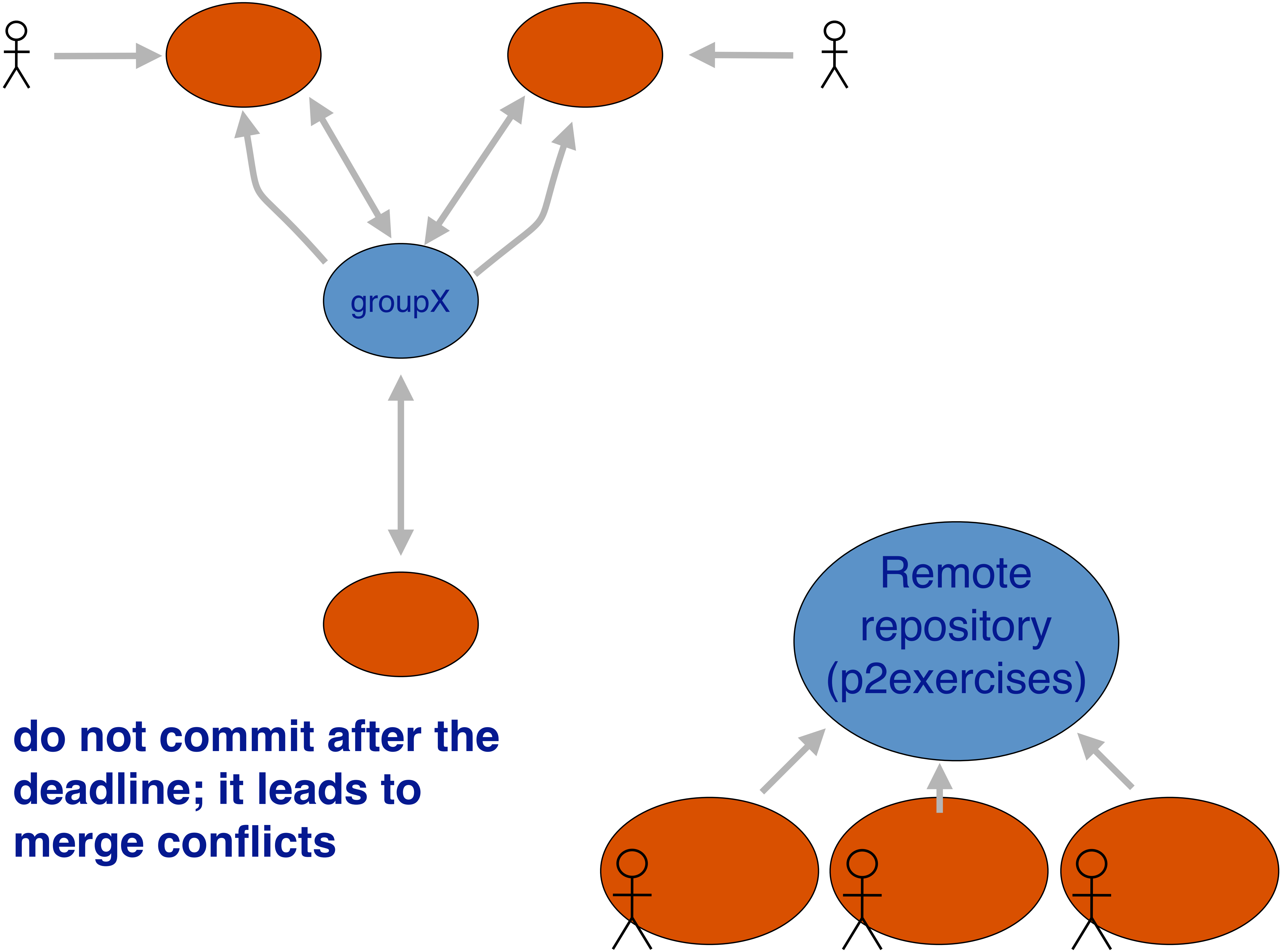
```
git remote add p2exercises ...

git pull p2exercises master
```



groupX



Remote repository (p2exercises)



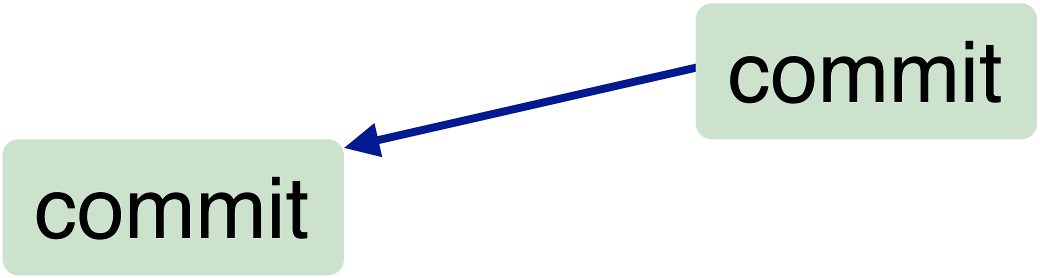
do not commit after the deadline; it leads to merge conflicts

Basic git

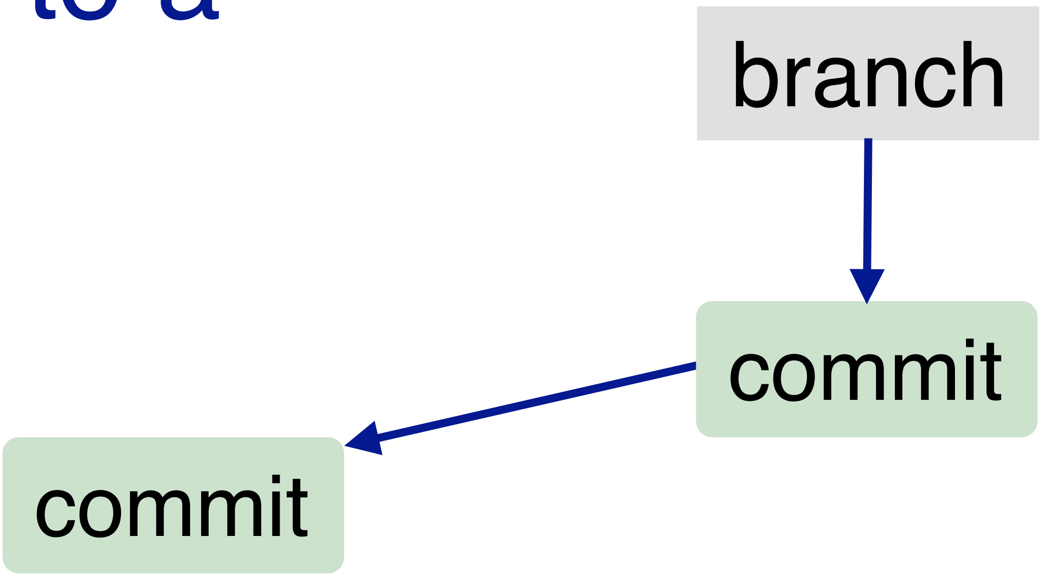
A “commit” is
“a set of changes”
to a “set of files”

commit

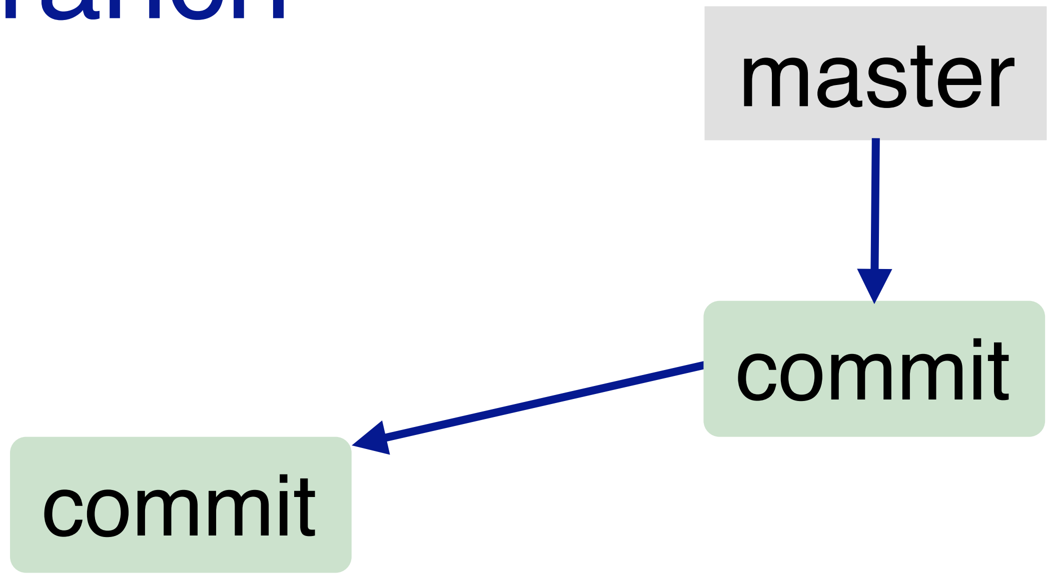
Most commits modify (or merge) earlier commits



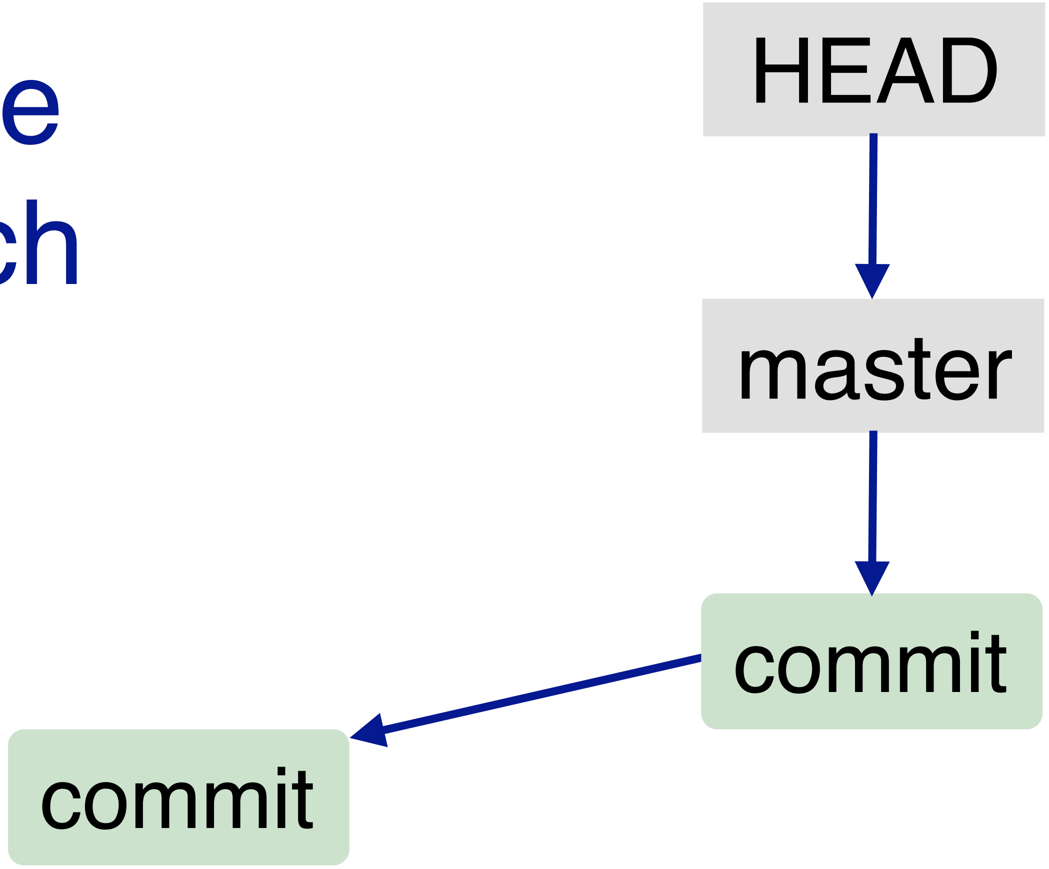
A graph of commits may belong to a *branch*



master
is the main branch

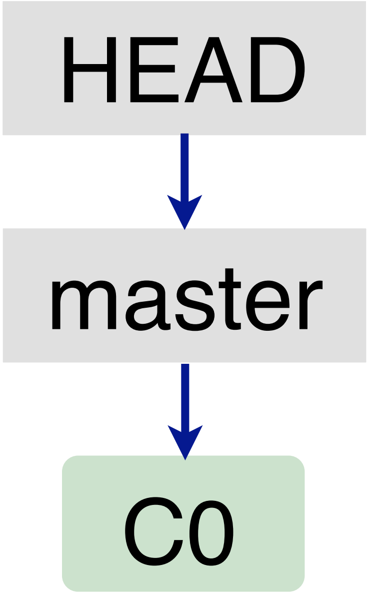


“HEAD “is the current branch



Create a git repo

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



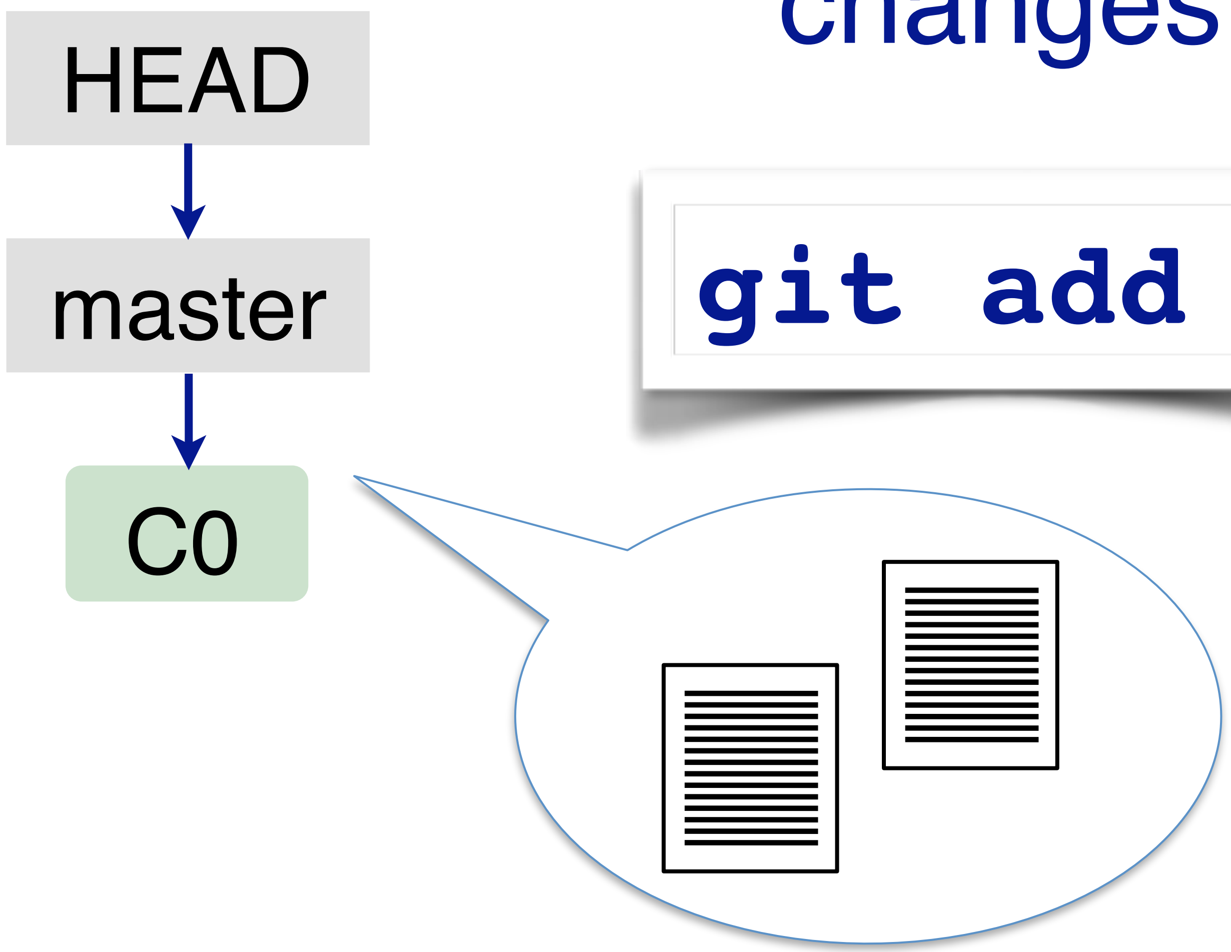
HEAD

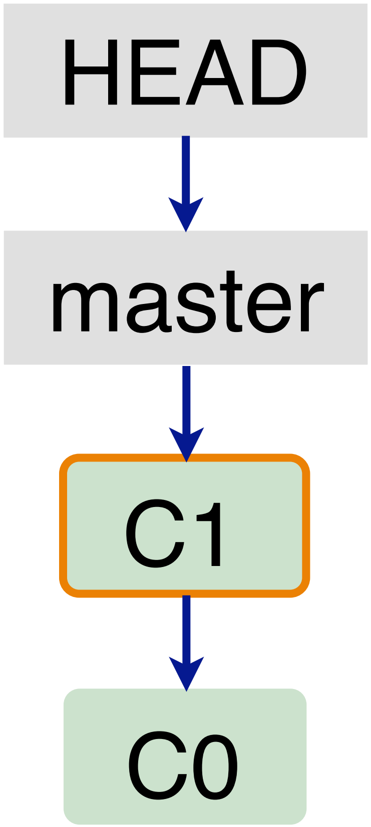
master

C0

Tell git to “stage” changes

```
git add …
```





HEAD

master

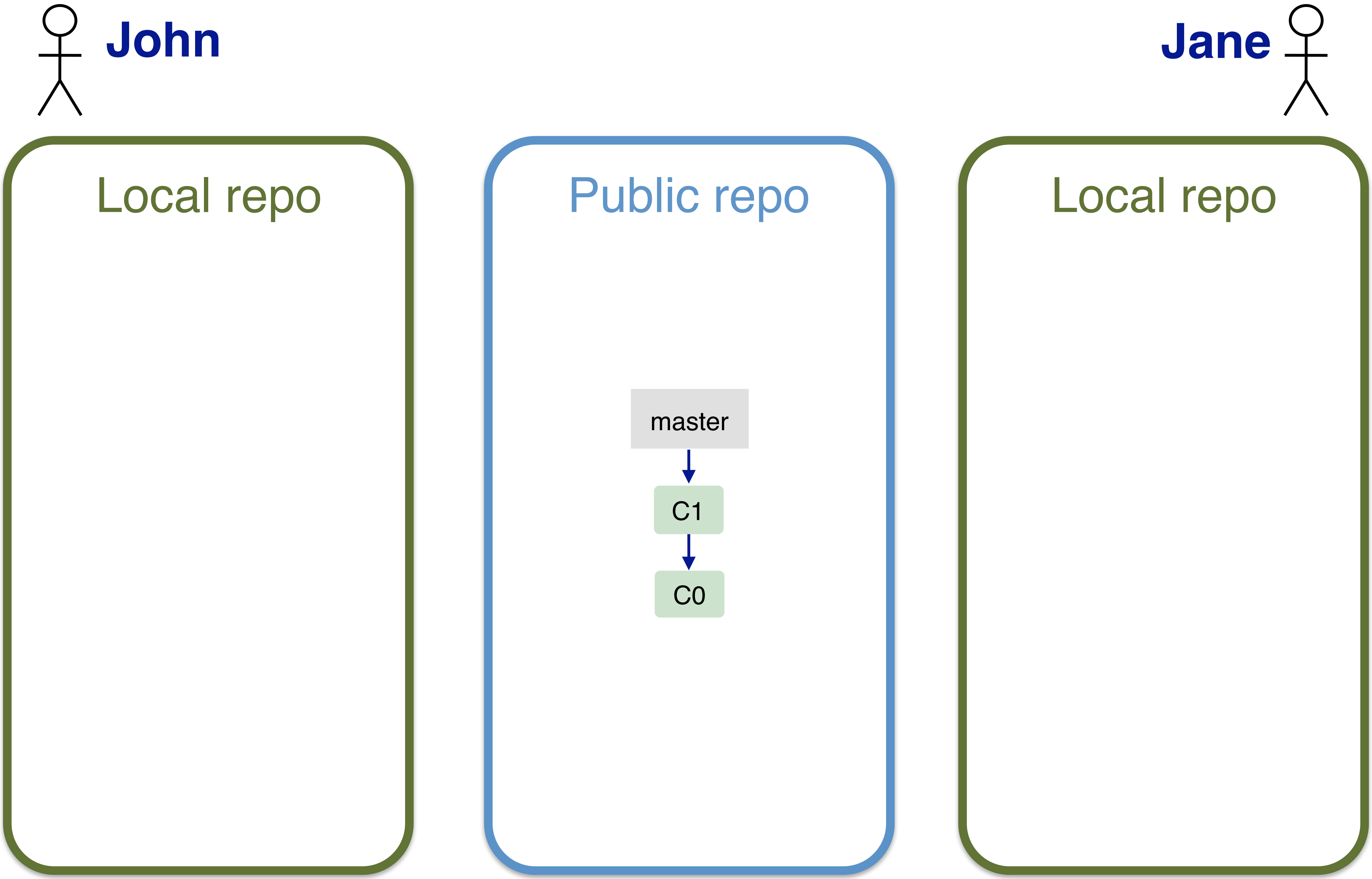
C1

C0

Commit your changes

```
git commit …
```

Collaborating



John

Jane

Local repo

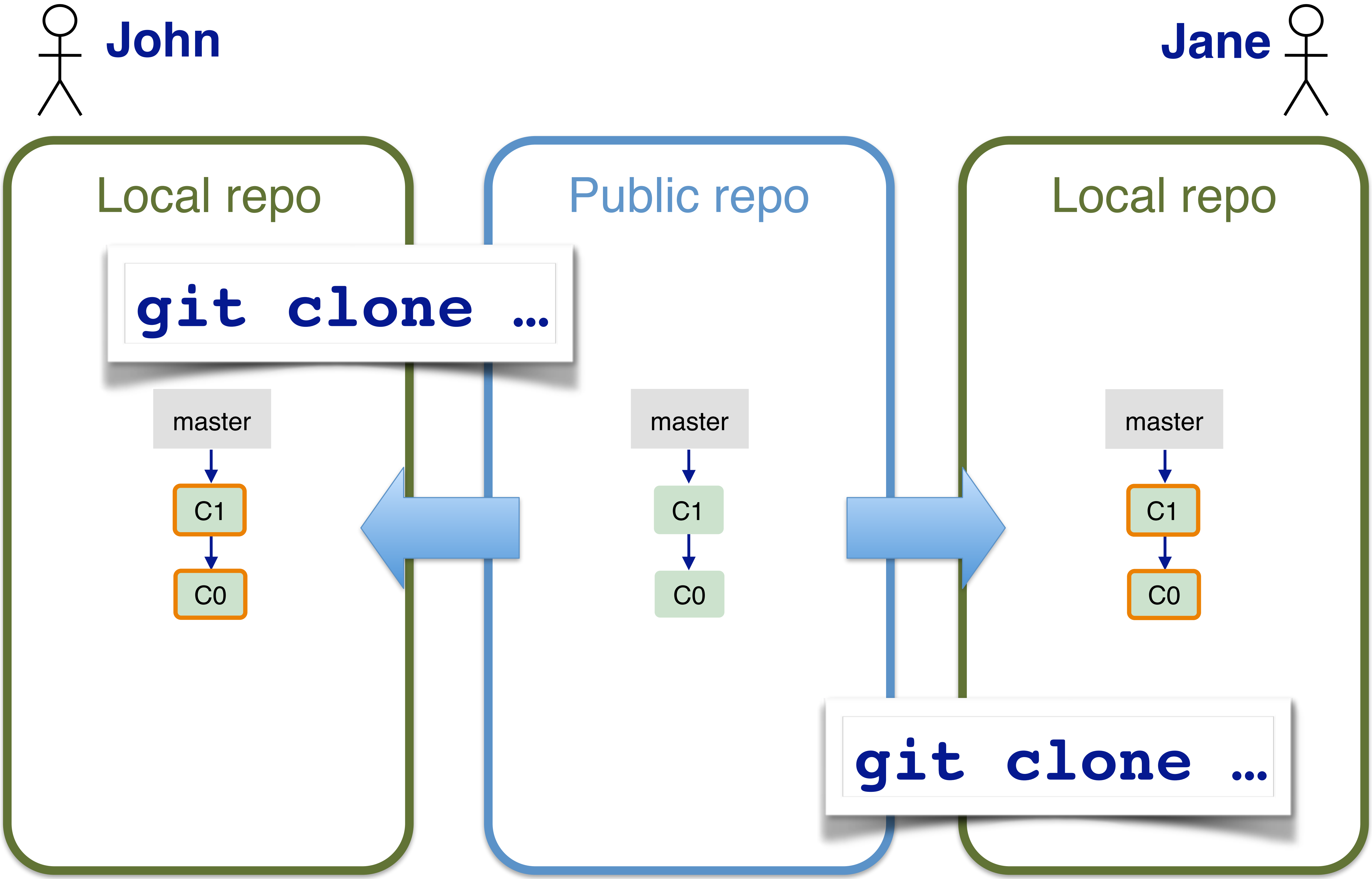
Public repo

Local repo

master

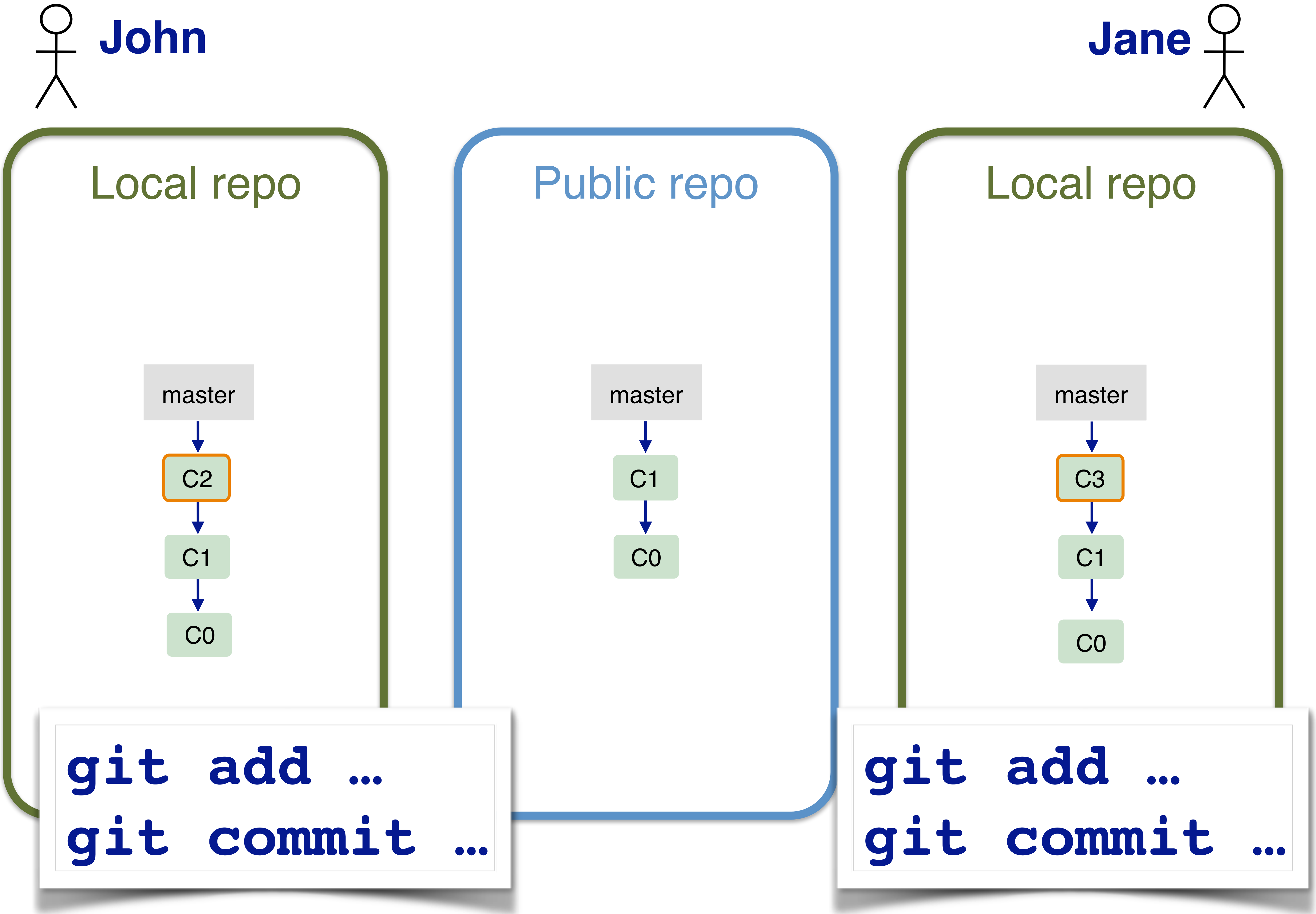
C1

C0



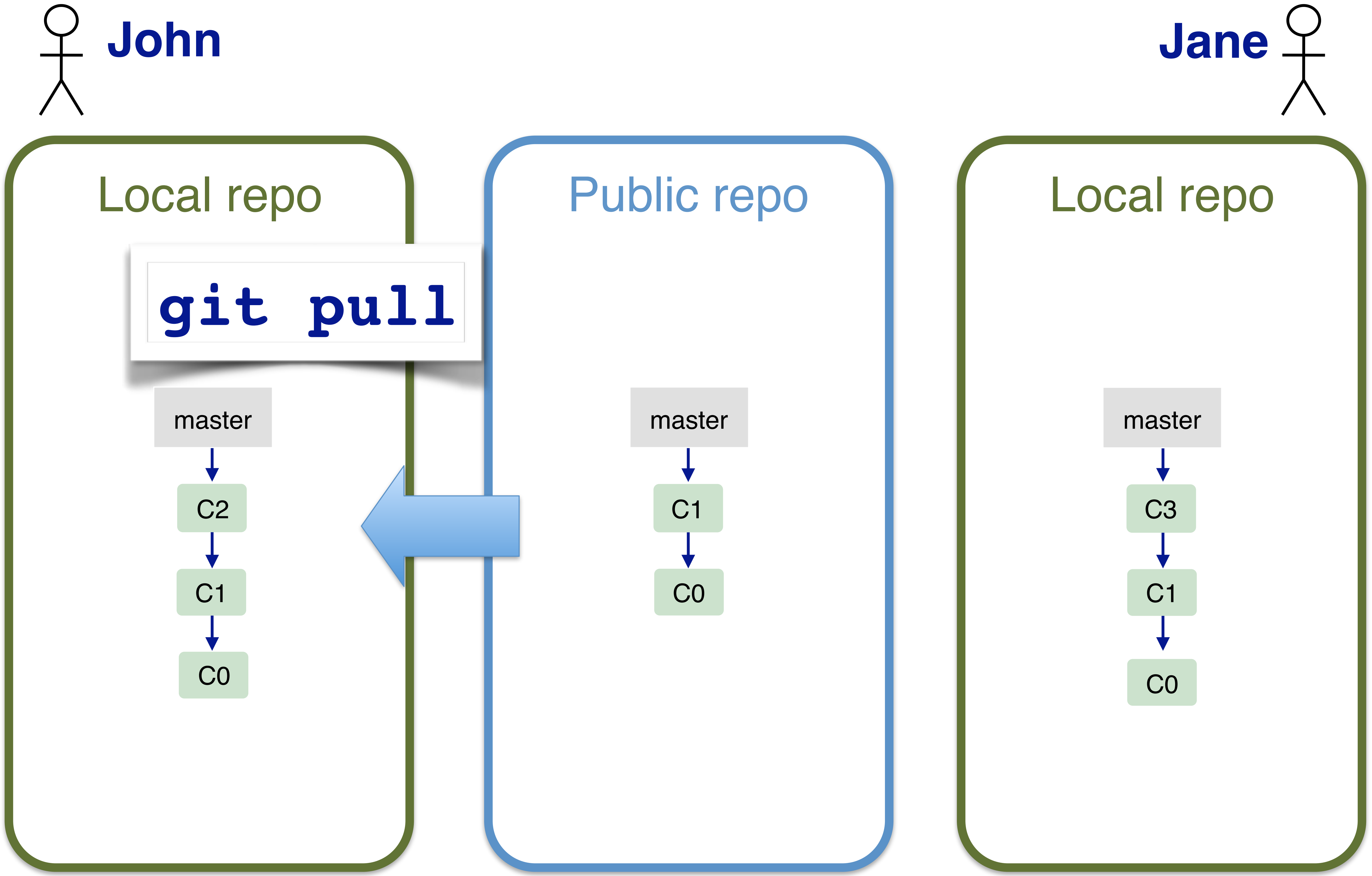
```
git clone …
```

```
git clone …
```



```
git add …
git commit …
```

```
git add …
git commit …
```



John

Jane

Local repo

Public repo

Local repo

```
git pull
```

master

C2

C1

C0

master

C1

C0

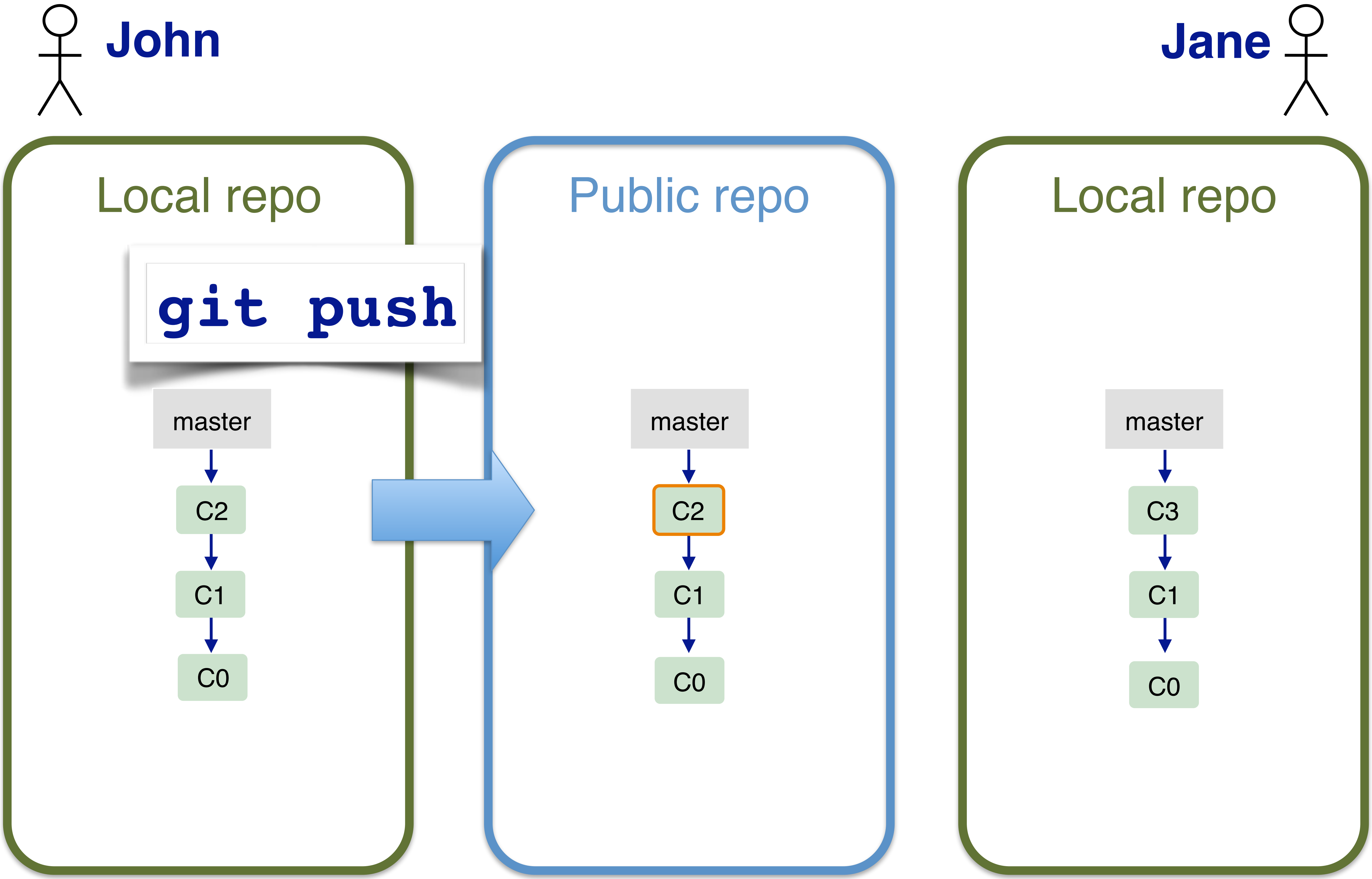
master

C3

C1

C0

(nothing new to pull)



John

Jane

Local repo

```
git push
```

master

C2

C1

C0

Public repo

master

C2

C1

C0

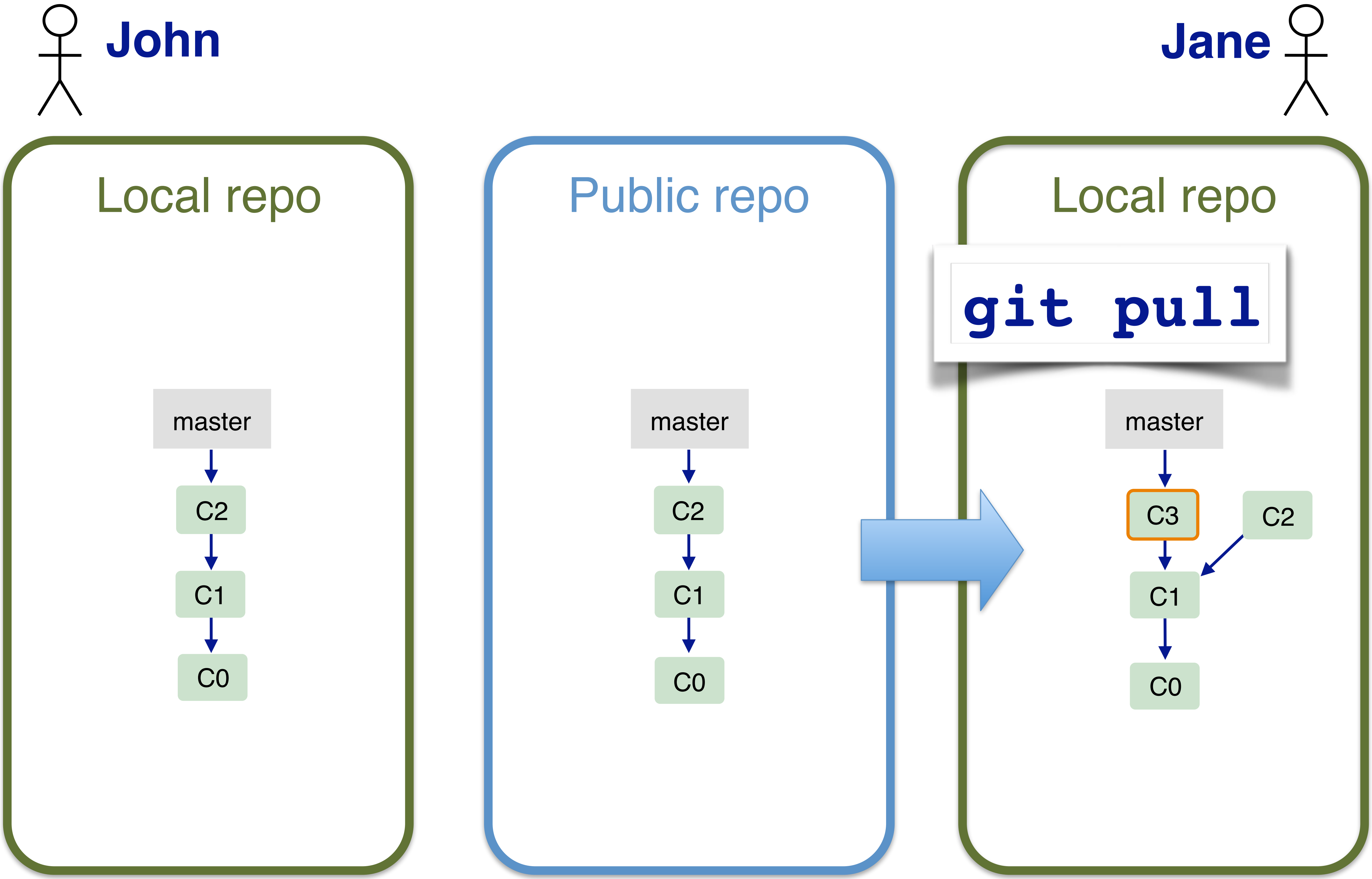
Local repo

master

C3

C1

C0



John

Jane

Local repo

master

C2

C1

C0

Public repo

master

C2

C1

C0

Local repo

```
git pull
```

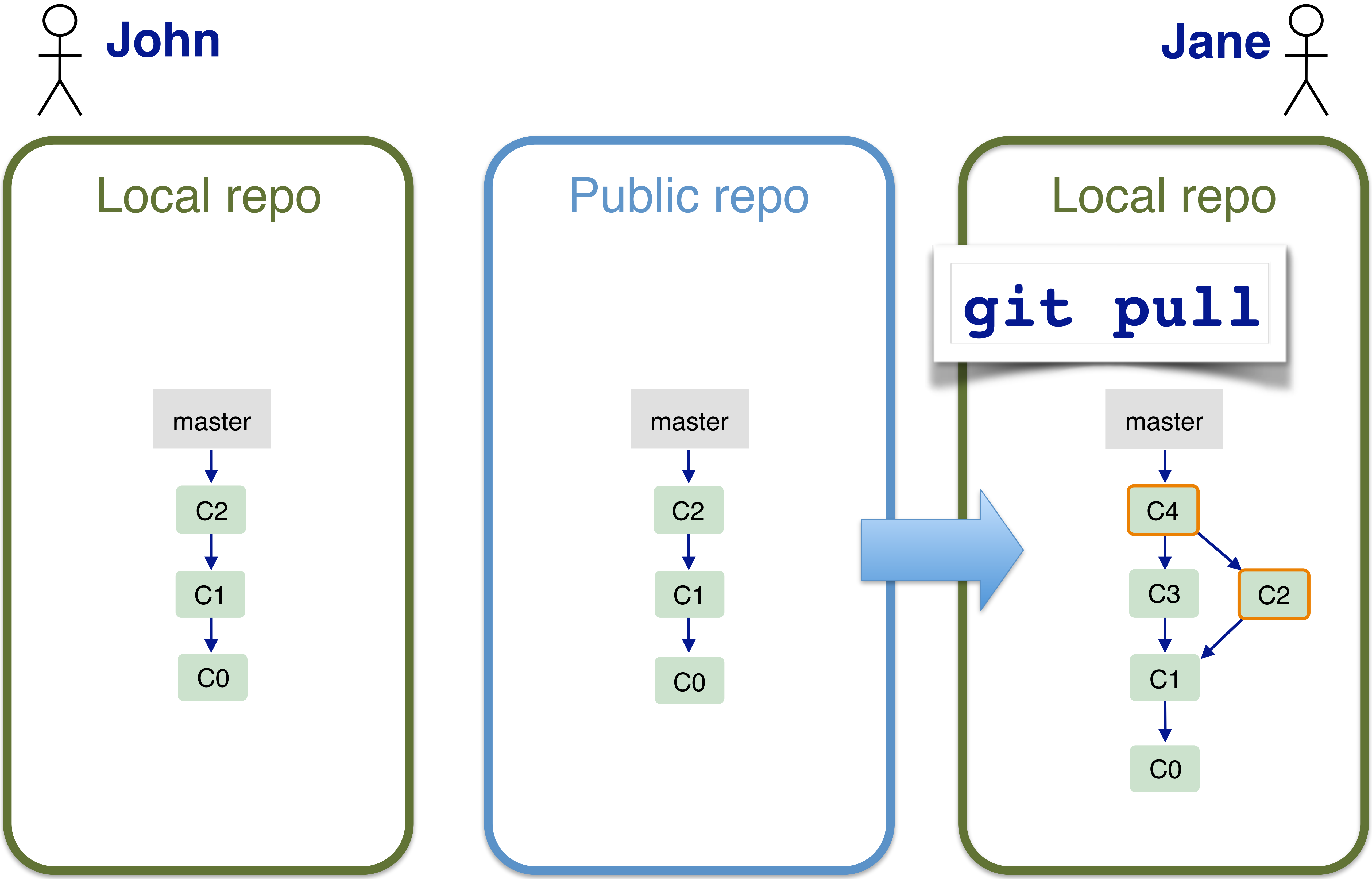
master

C3

C2

C1

C0



John

Jane

Local repo

master

C2

C1

C0

Public repo

master

C2

C1

C0

Local repo

```
git pull
```

master

C4

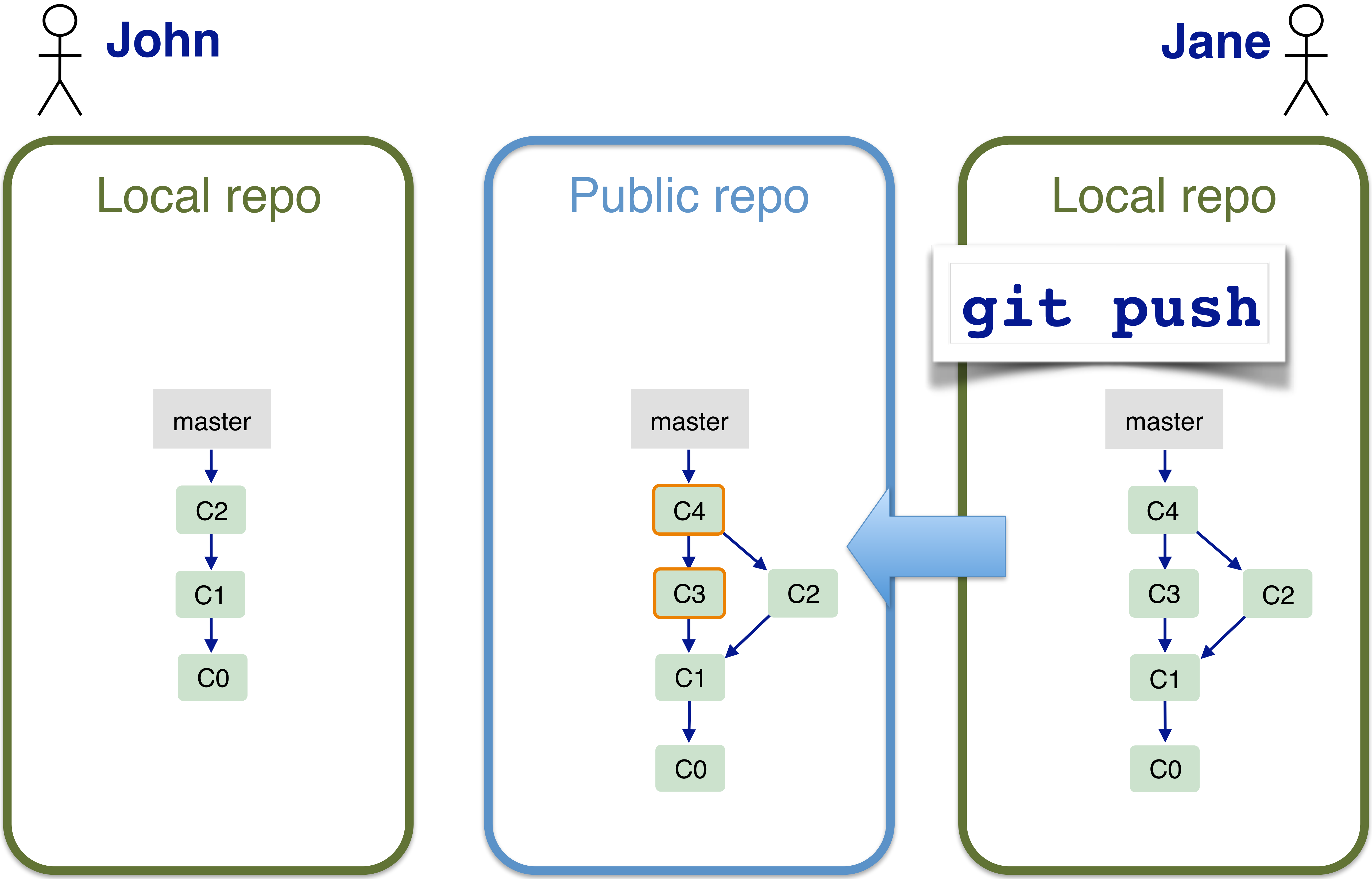
C3

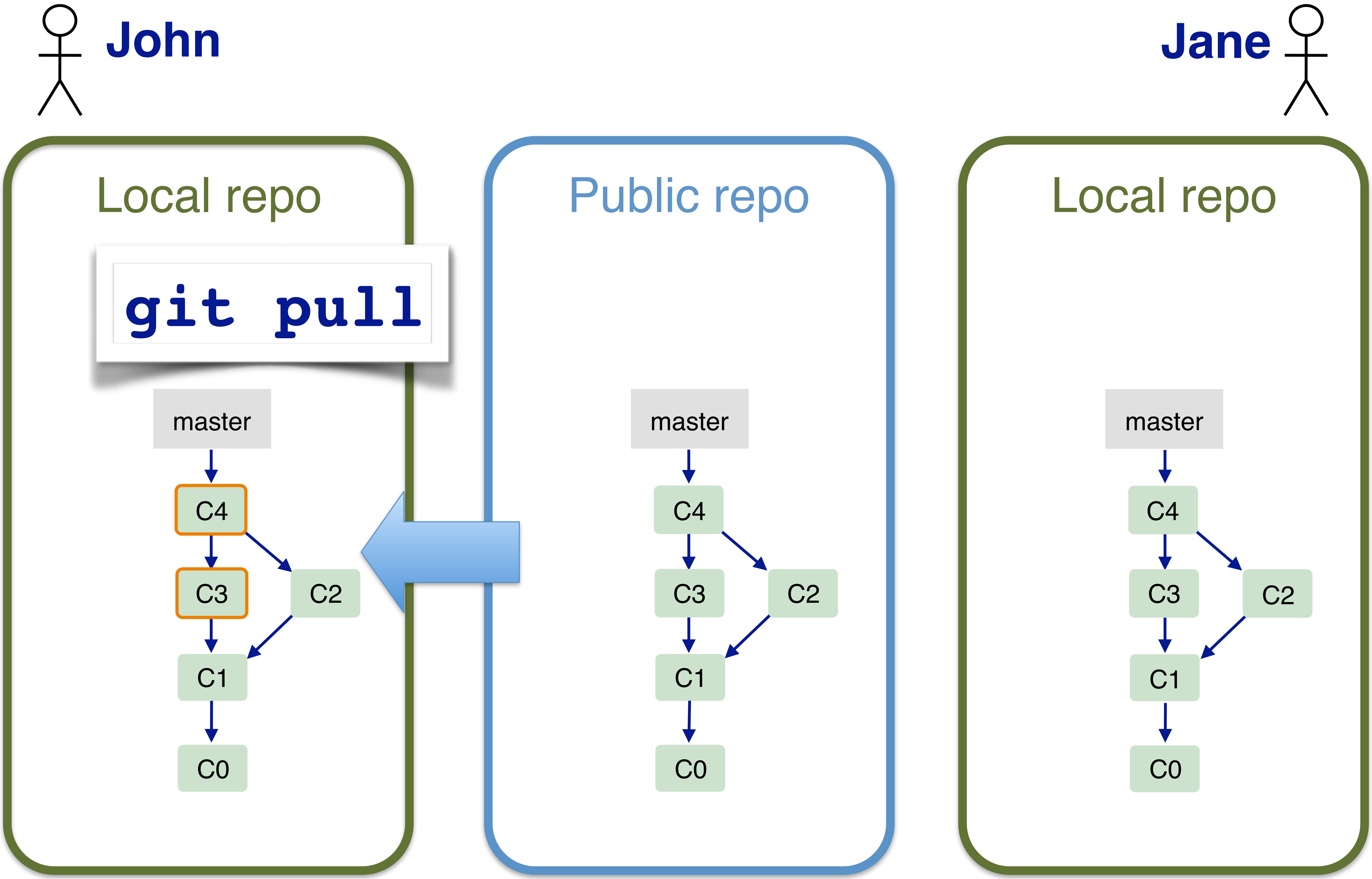
C2

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
```

In case of fire

1. git commit
2. git push
3. leave building

to be continued

Resources

http://git-scm.com/

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

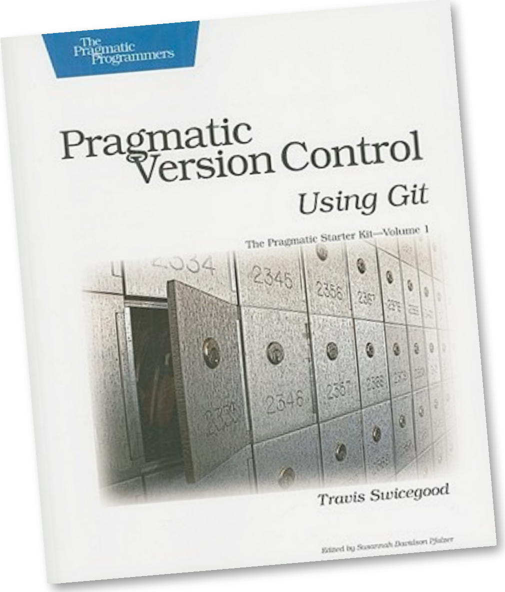
github
SOCIAL CODING

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

Edited by Susannah Davidson Pfalzer

http://oreilly.com/