
OREO

Version Control Using Git

Lab 3 – CSCI 3308 – Spring 2017

1. git commit

2. git push

3. leave building

Slide by Oscar Nierstrasz

Version Control – Lingo

Setup

- Source Control
- Repository
- Server
- Client
- Working Copy

Actions

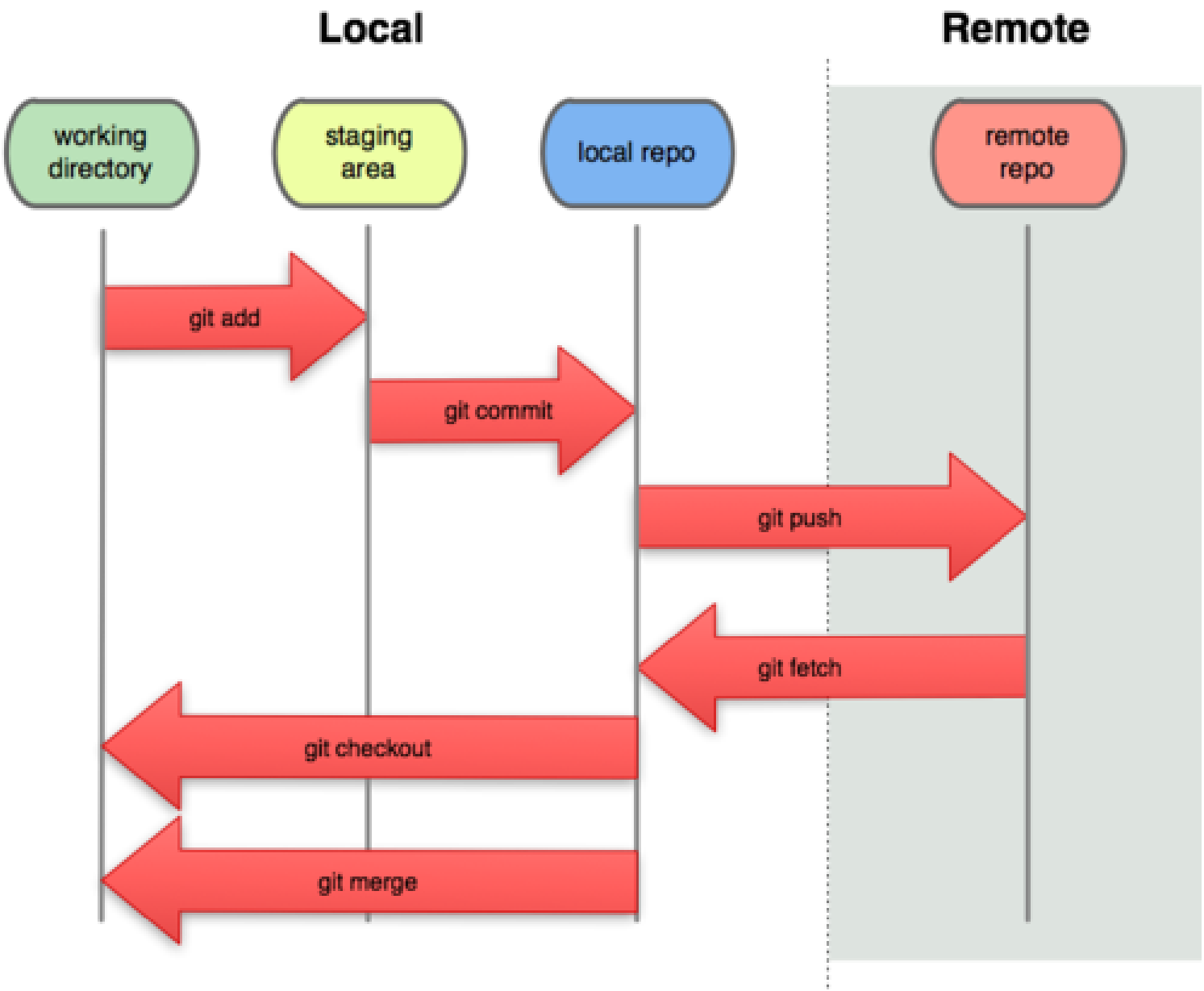
- Add
- Check out
- Check in / Commit
- Check in message
- Change log
- Update
- Revert

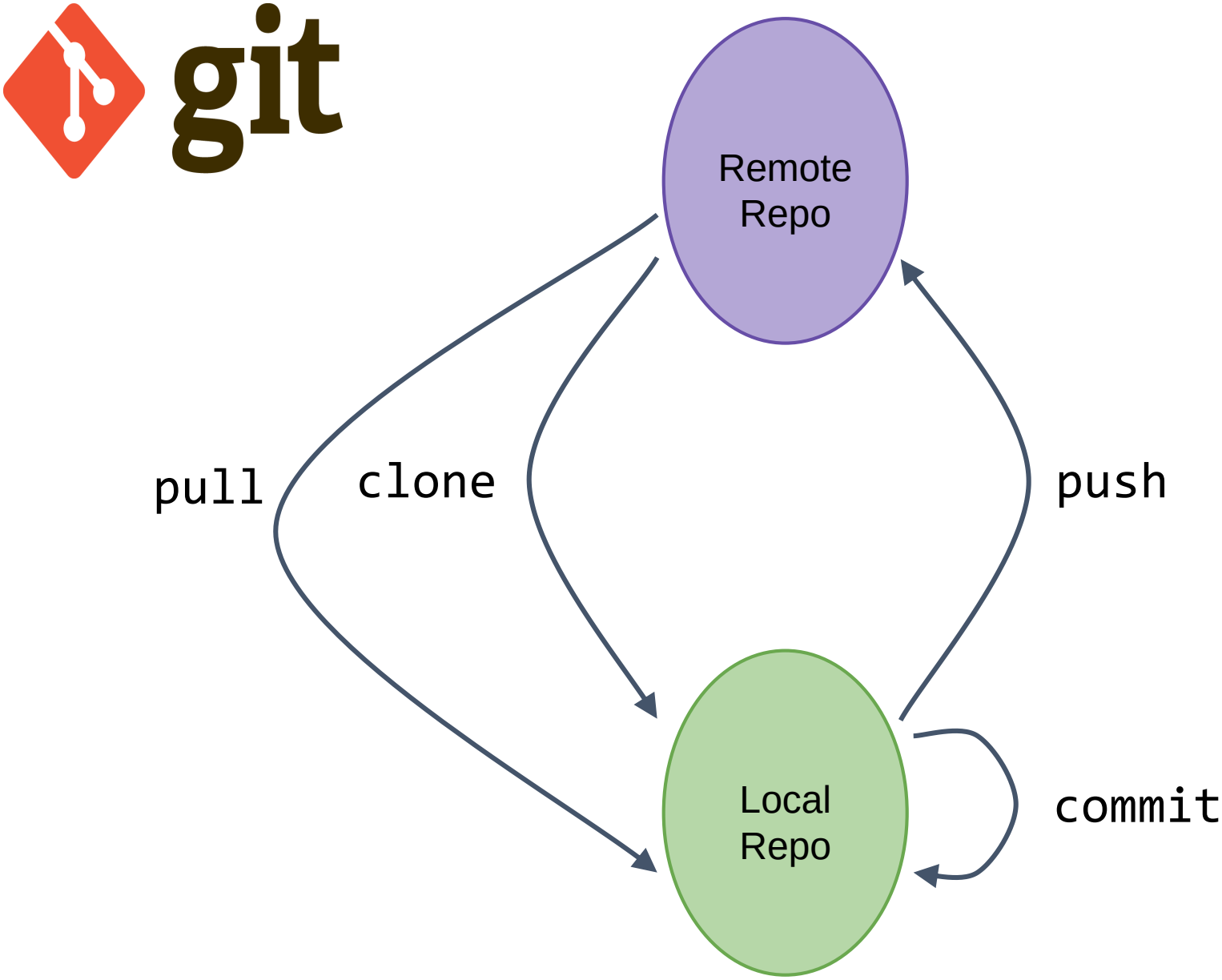
Advanced Actions

- Branch
- Diff
- Merge
- Conflict
- Resolve

git

Git Basics


Local
Remote
working directory
staging area
local repo
remote repo
git add
git commit
git push
git fetch
git checkout
git merge


git
Remote Repo
pull
clone
push
Local Repo
commit

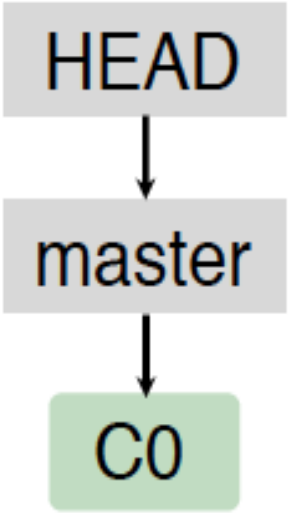
Git Workflows

From https://www.atlassian.com/git/workflows

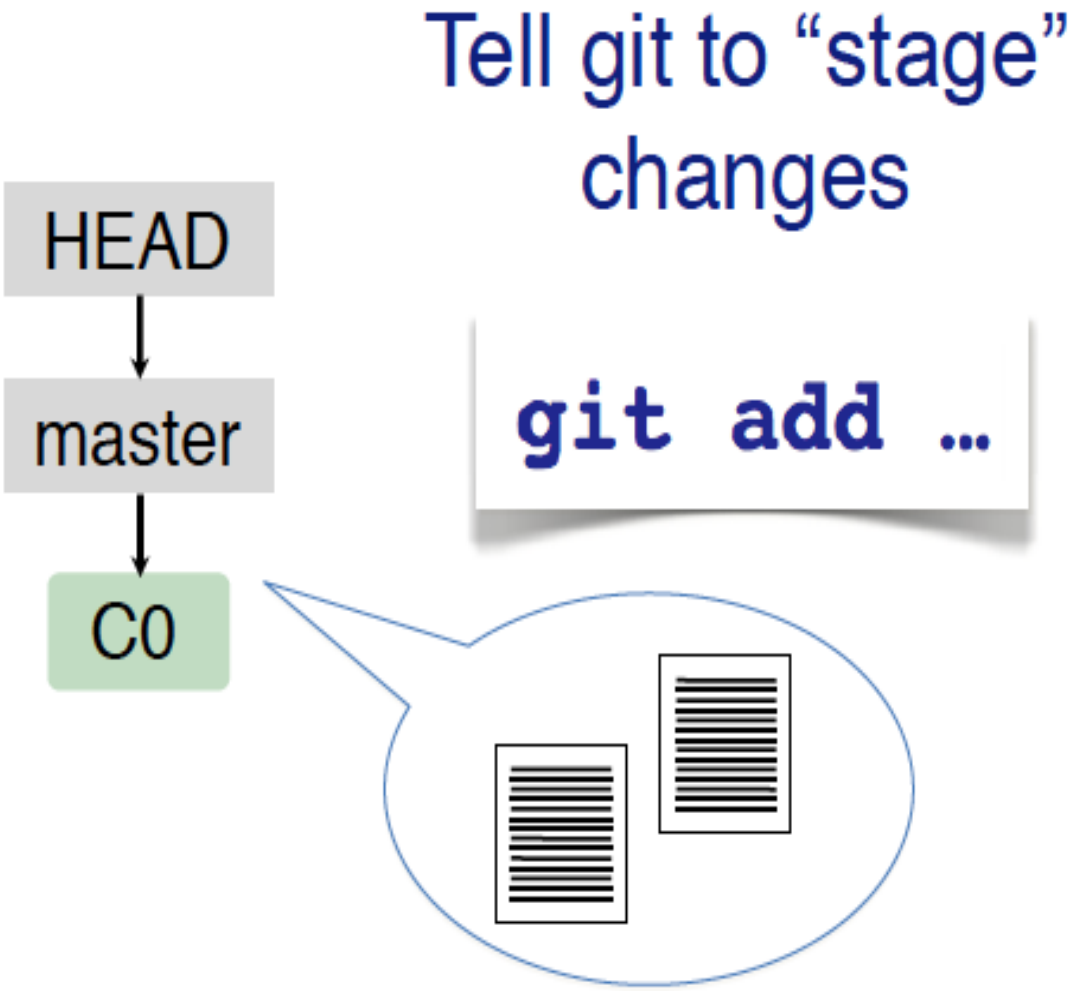
Workflows

Create a git repo

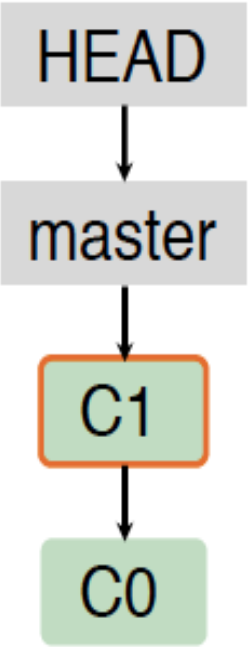
```
mkdir repo
cd repo
git init
```



Workflows



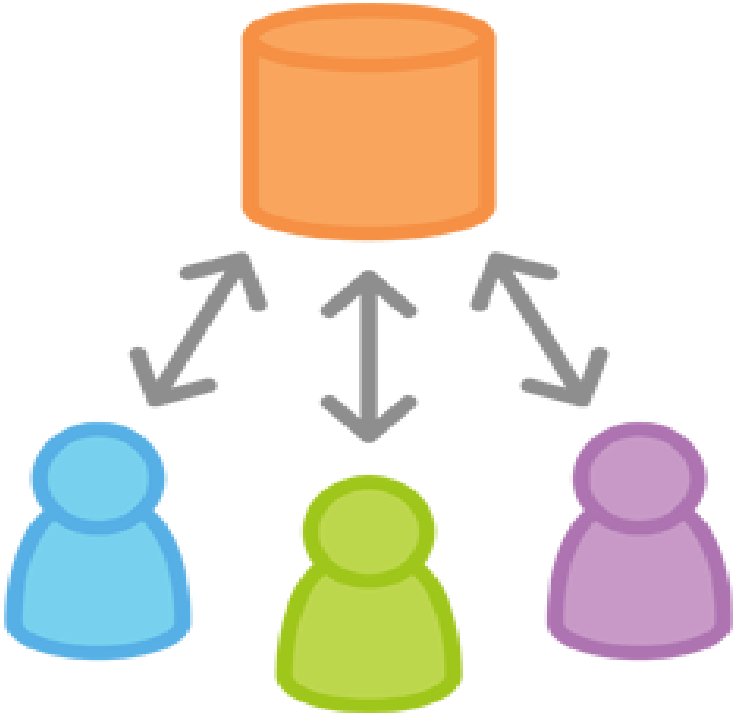
Workflows

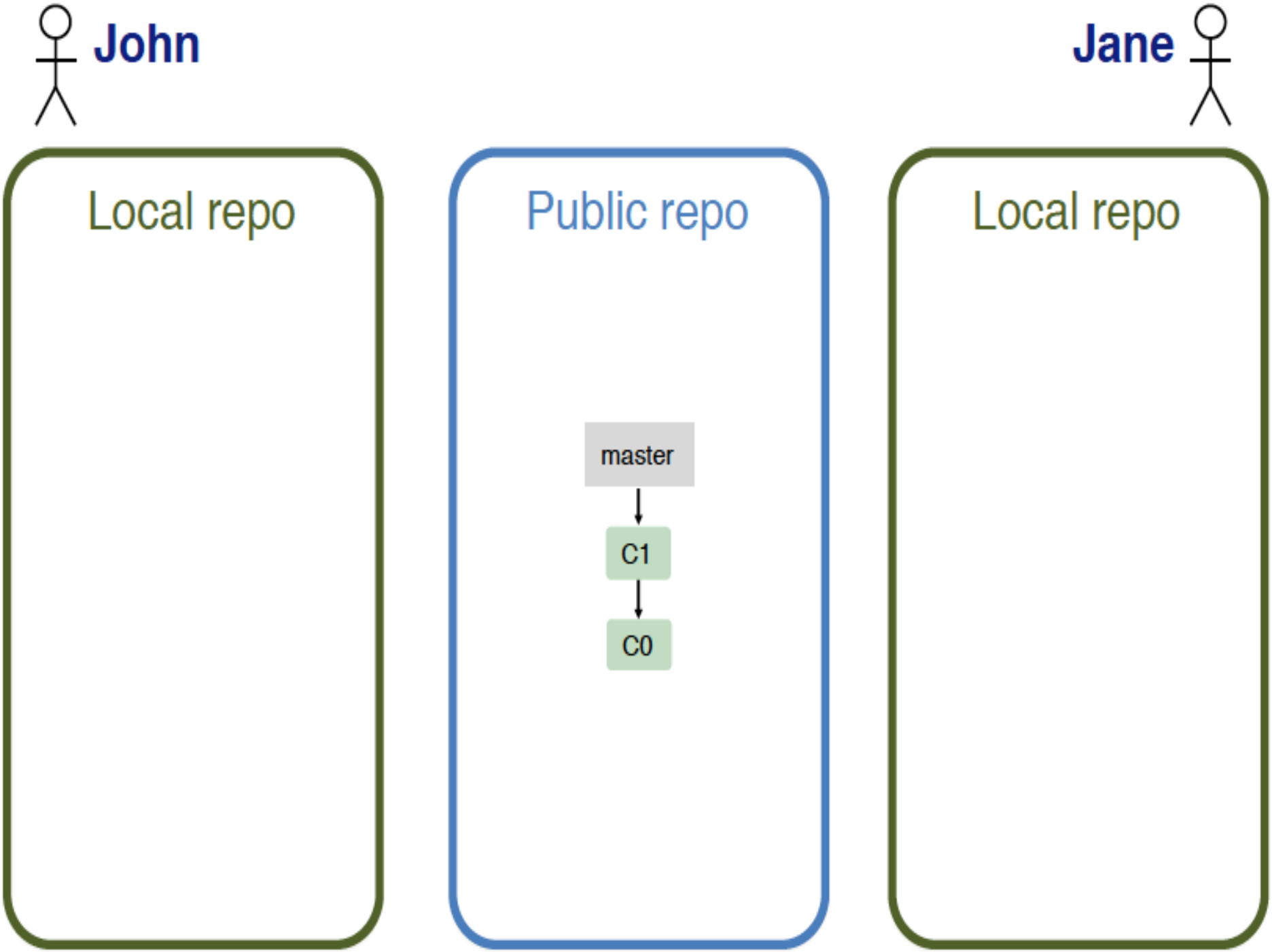


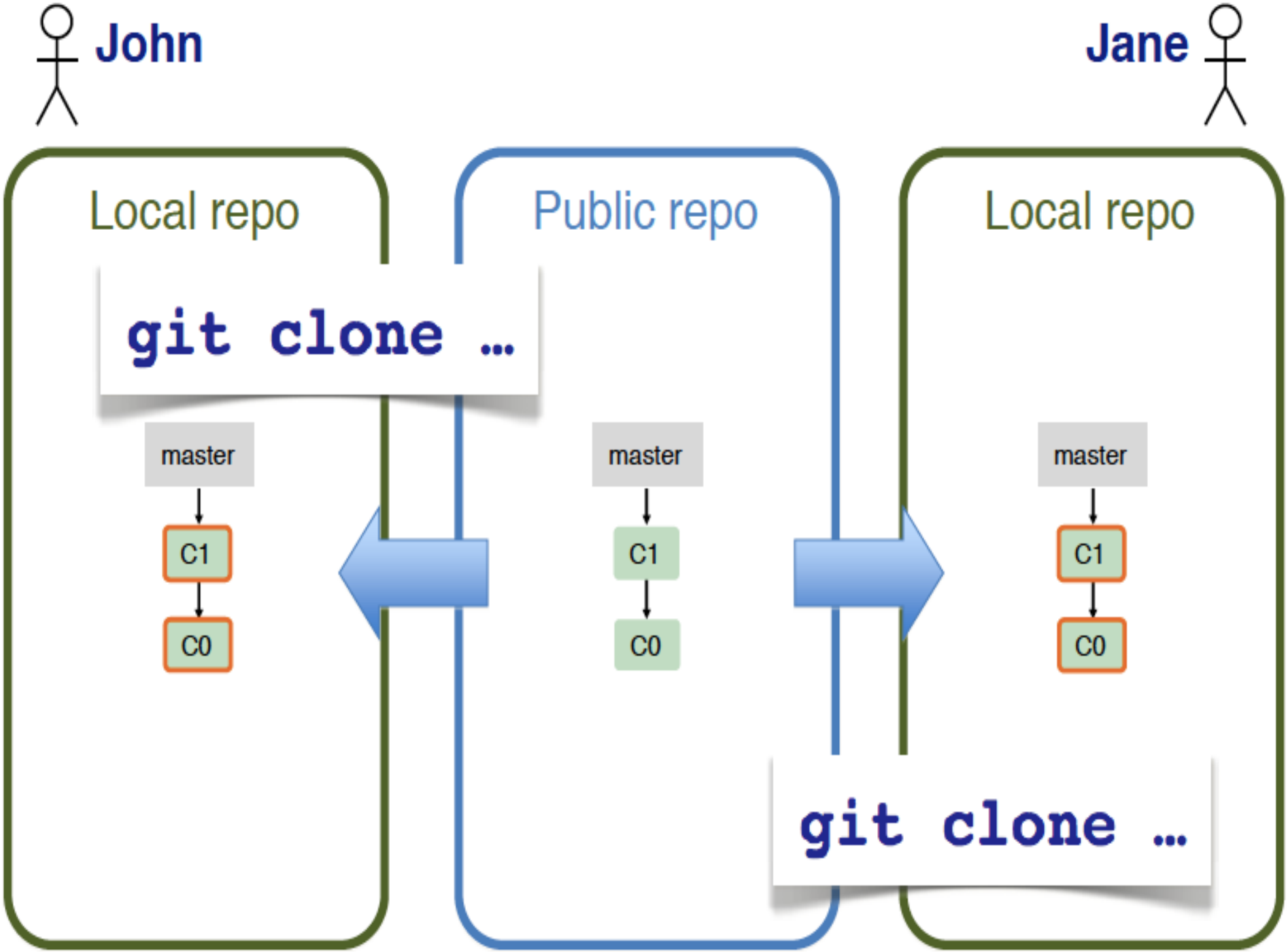
Commit your changes

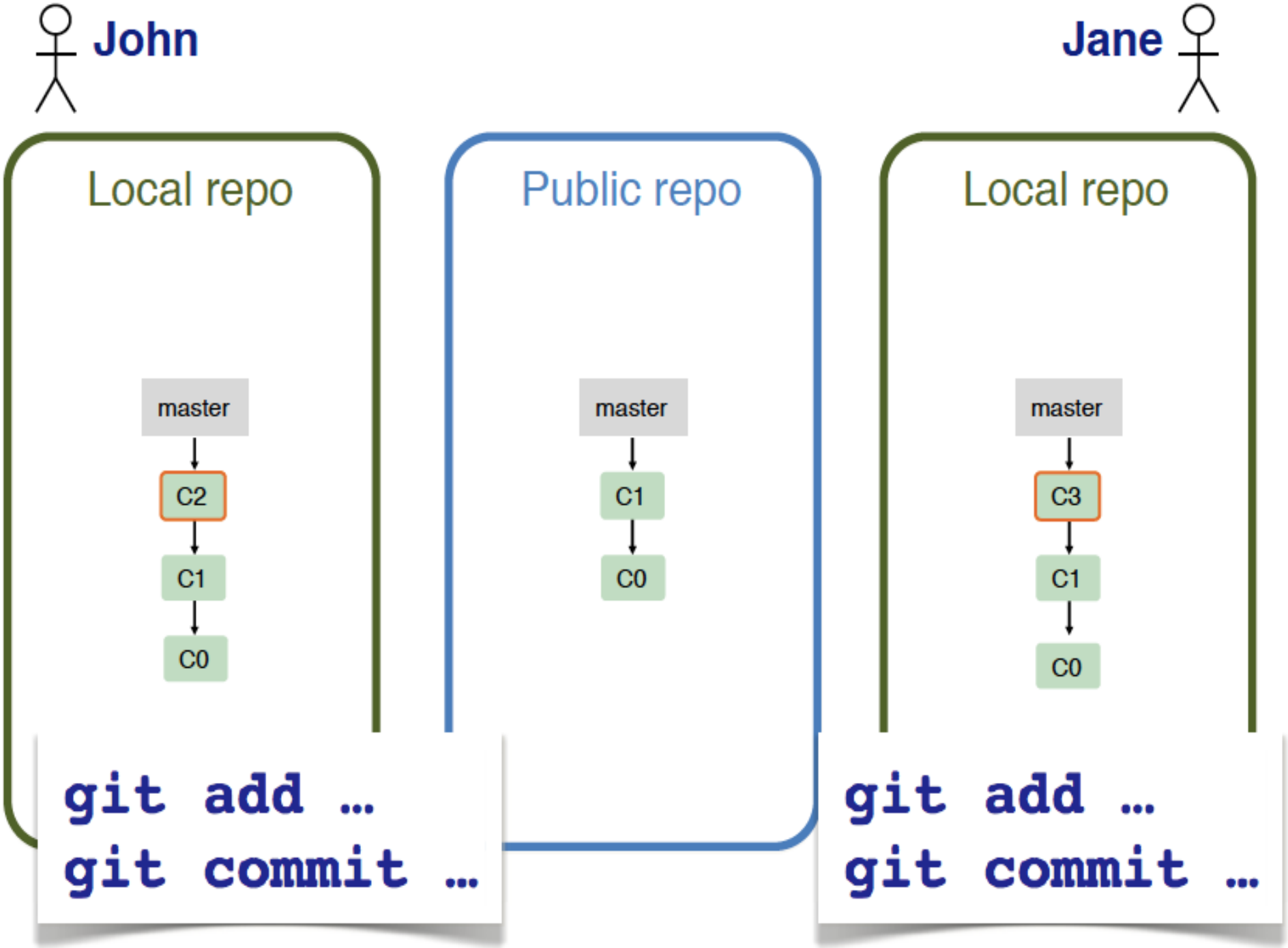
```
git commit …
```

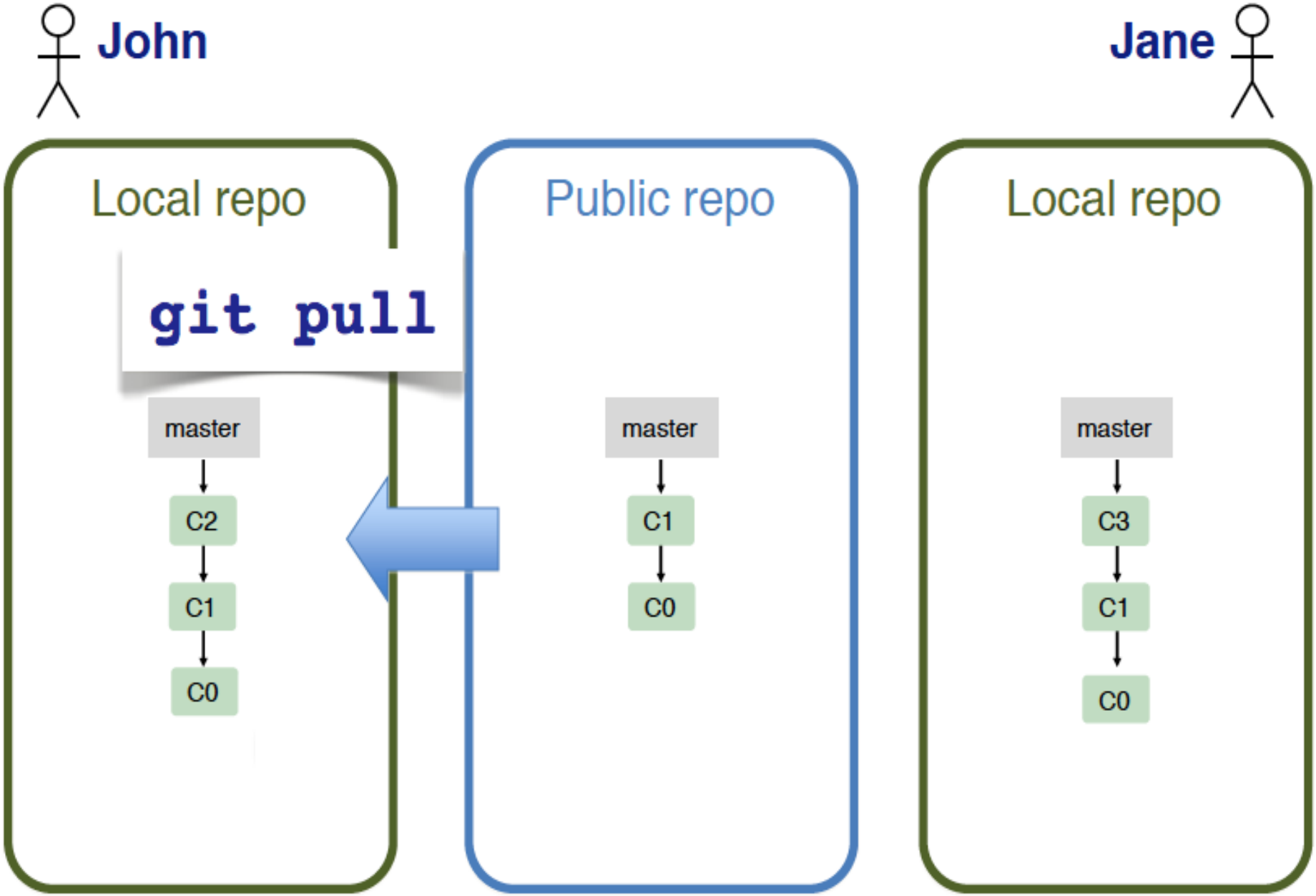
What happens when Collaborating?



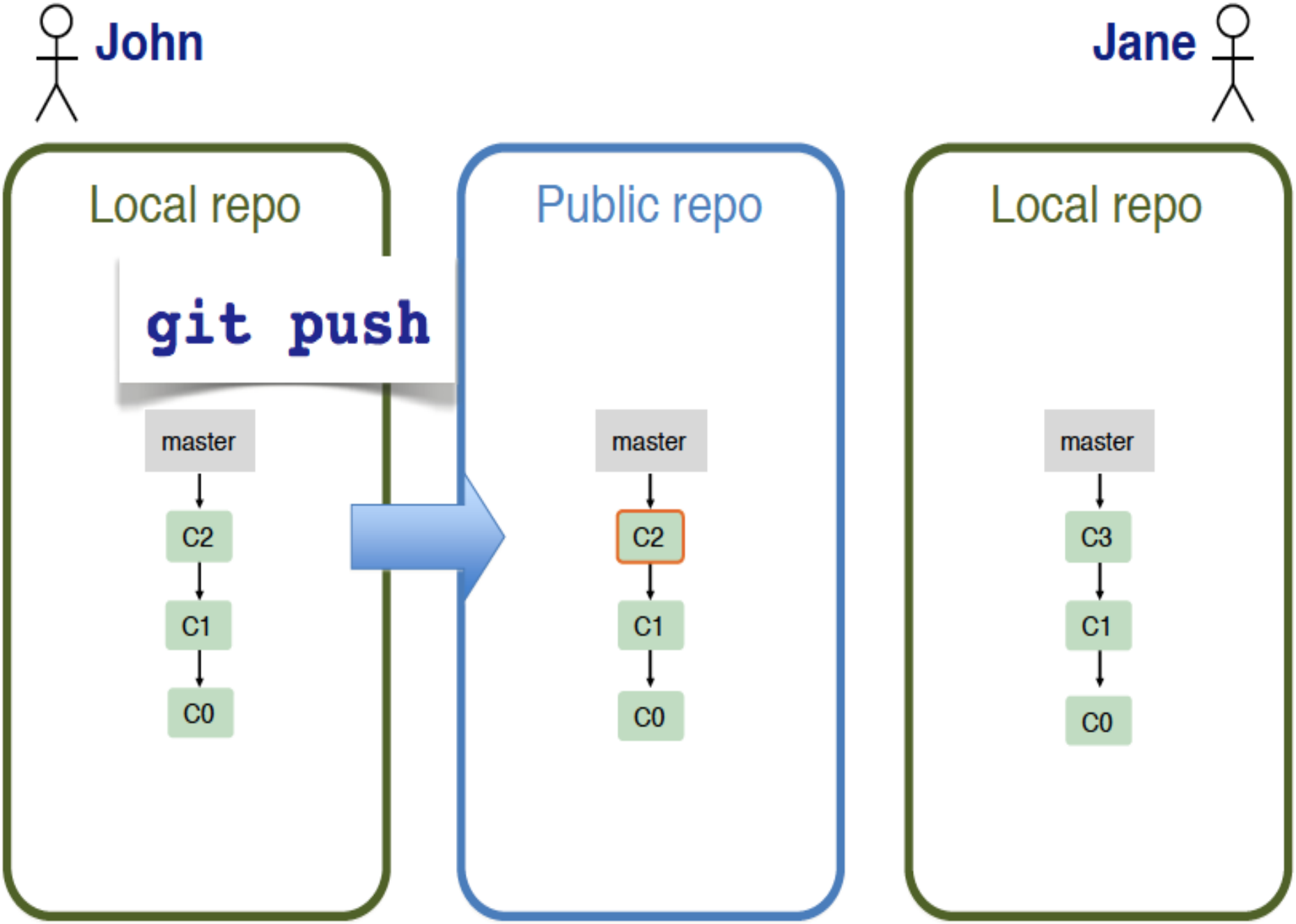

John
Jane
Local repo
Public repo
Local repo
master
C1
C0

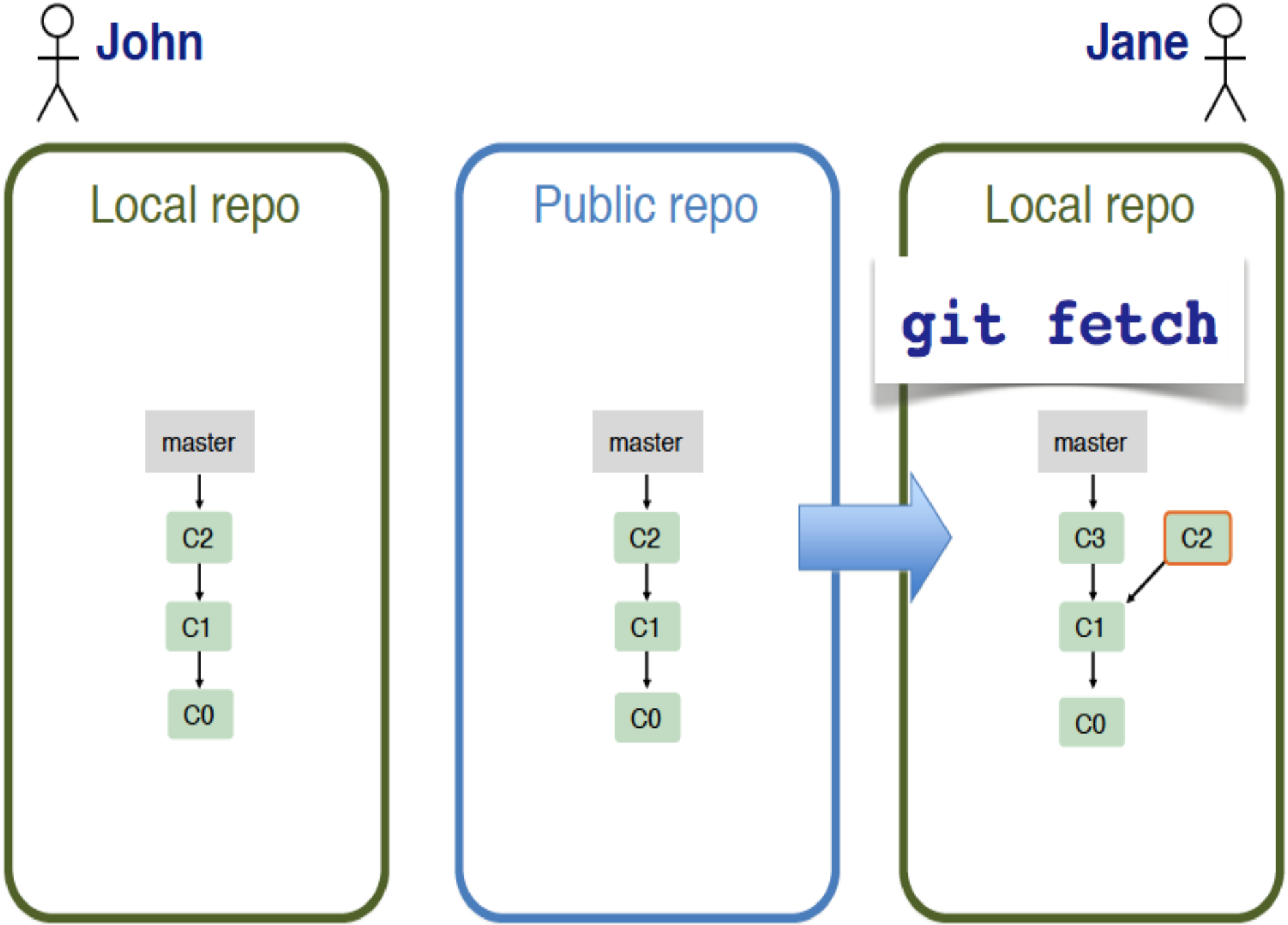

John
Jane
Local repo
Public repo
Local repo
git clone …
master
C1
C0
master
C1
C0
master
C1
C0
git clone …

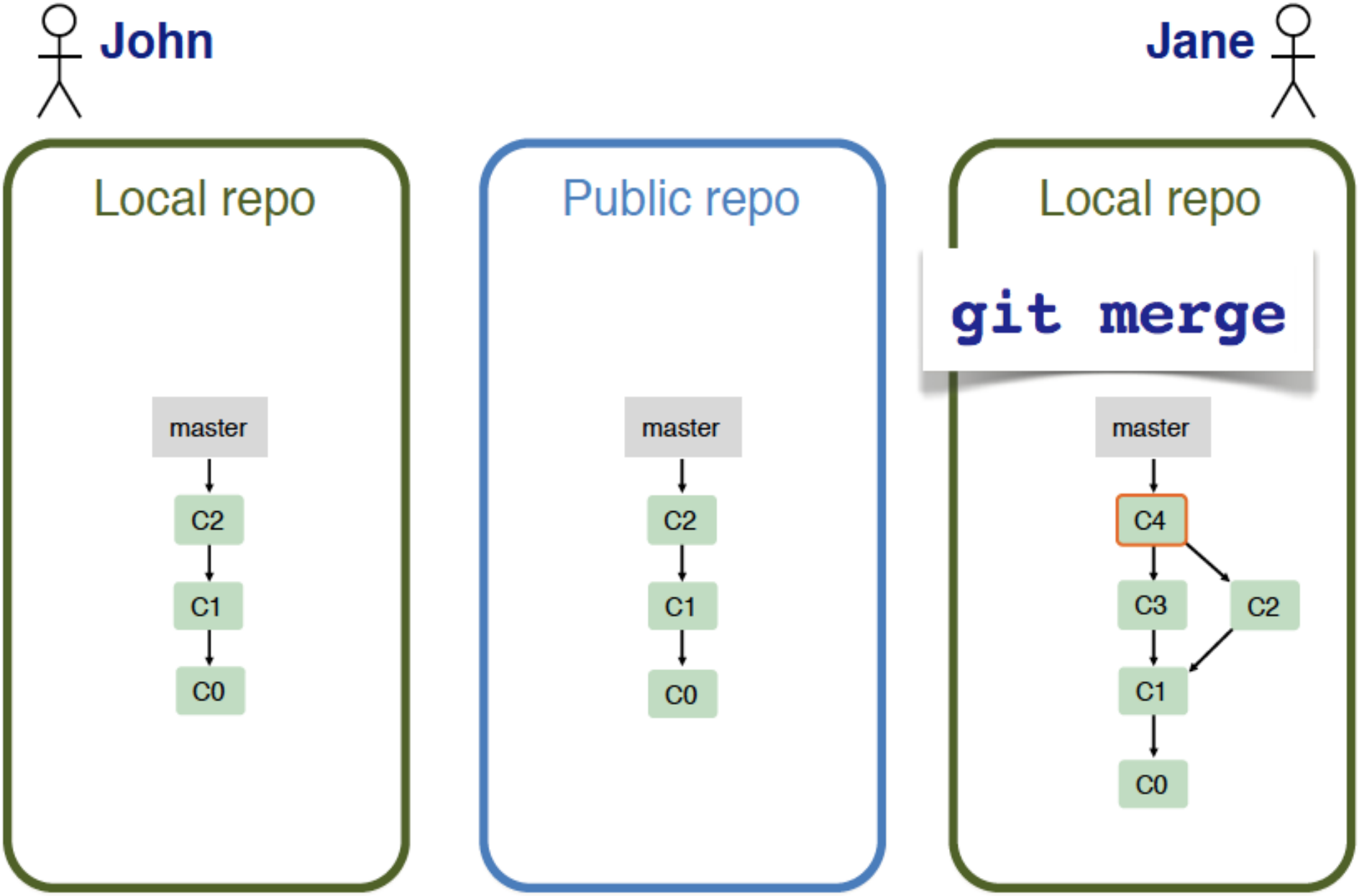

John
Jane
Local repo
Public repo
Local repo
master
C2
C1
C0
master
C1
C0
master
C3
C1
C0
git add …
git commit …
git add …
git commit …



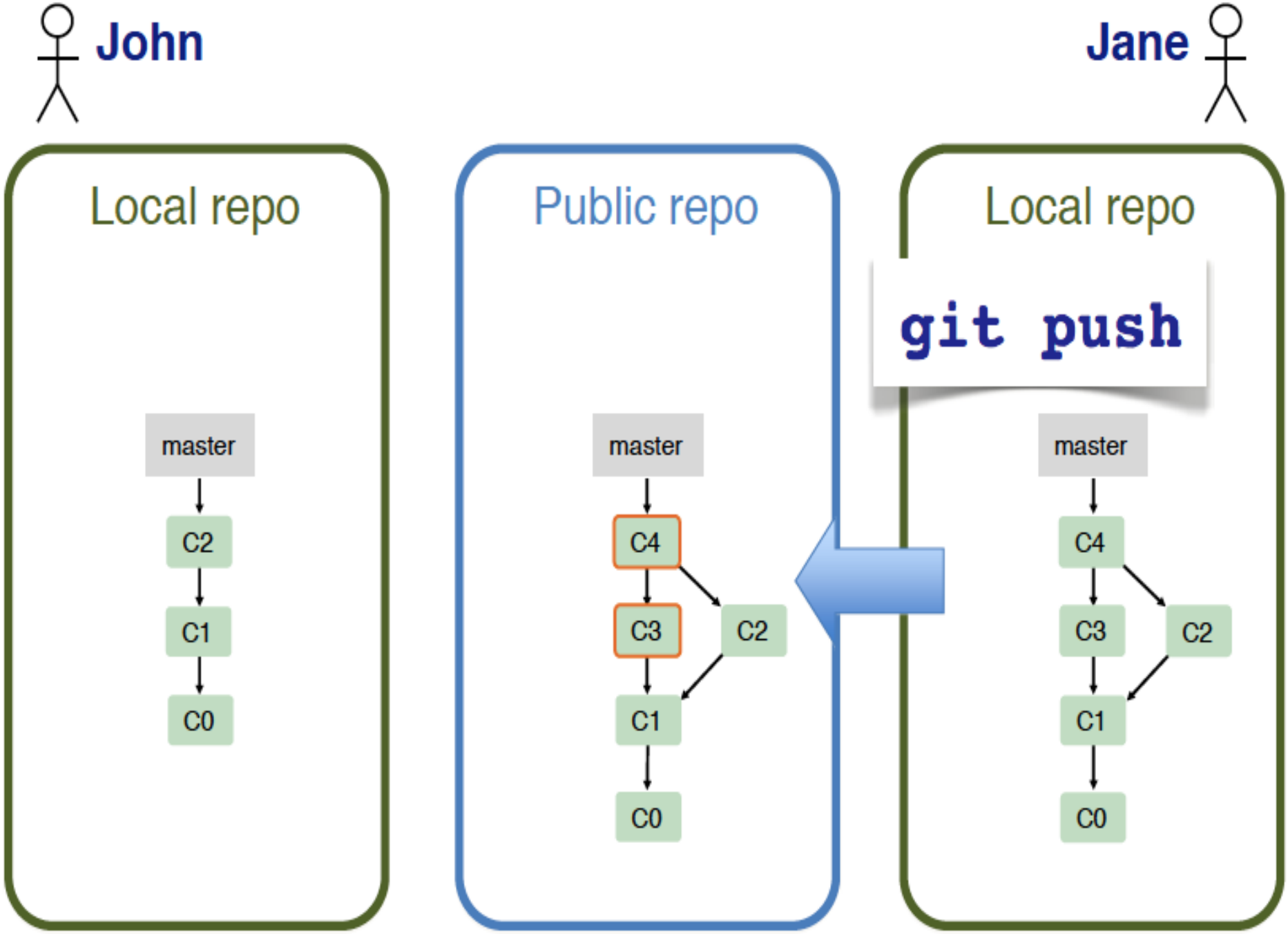
(nothing new to pull)

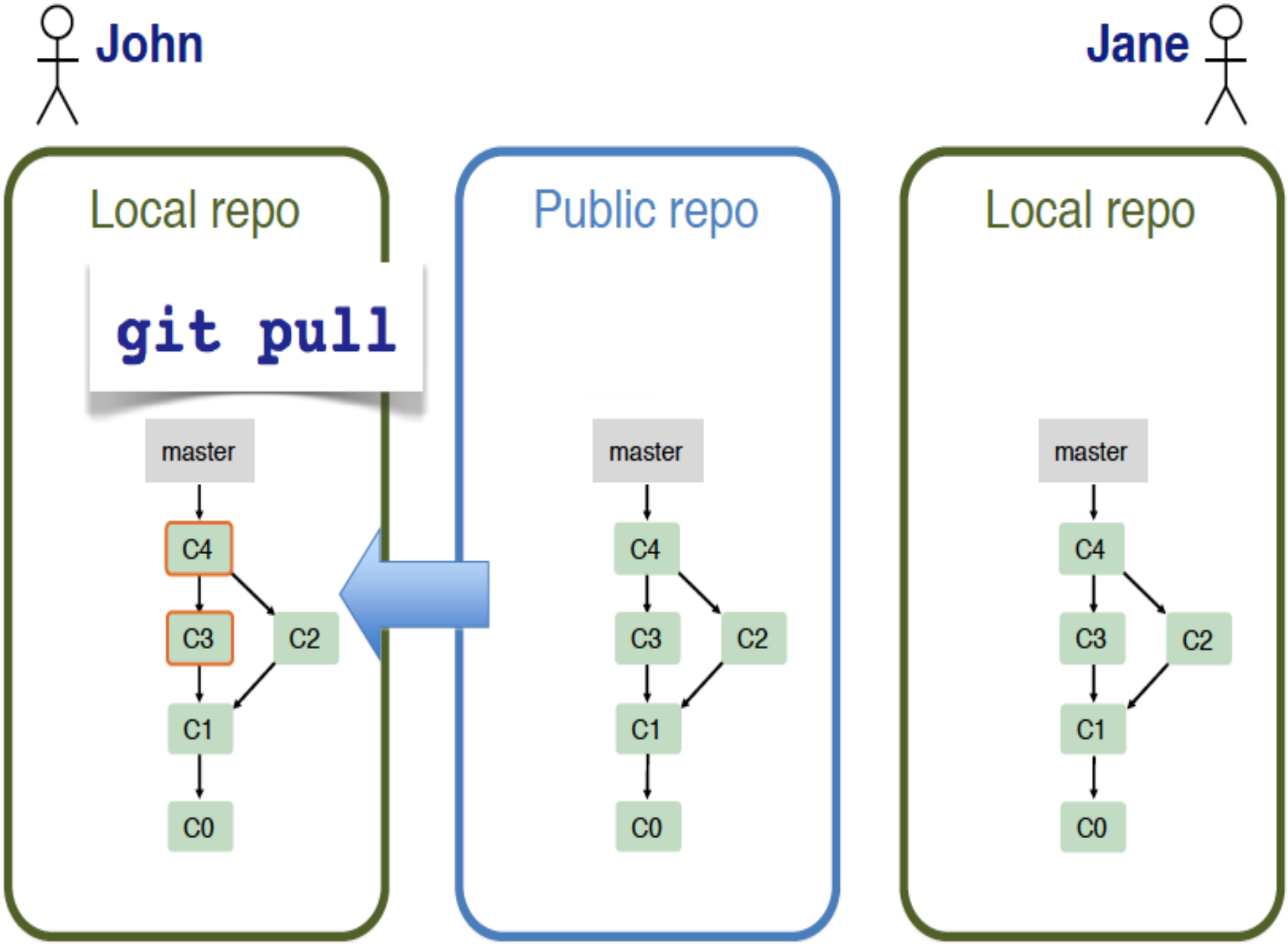

John
Jane
Local repo
git push
master
C2
C1
C0
Public repo
master
C2
C1
C0
Local repo
master
C3
C1
C0


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

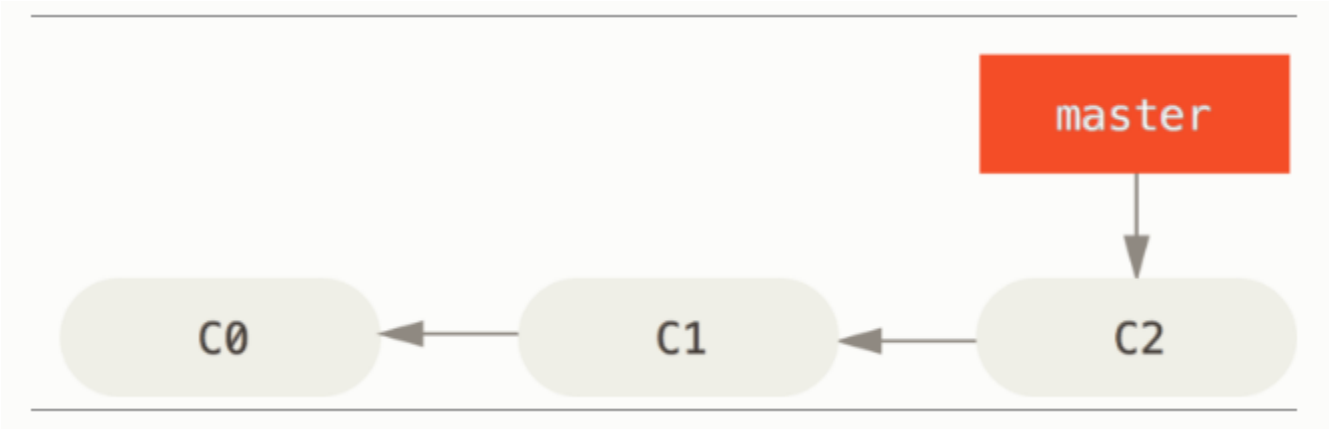
Branching & Merging

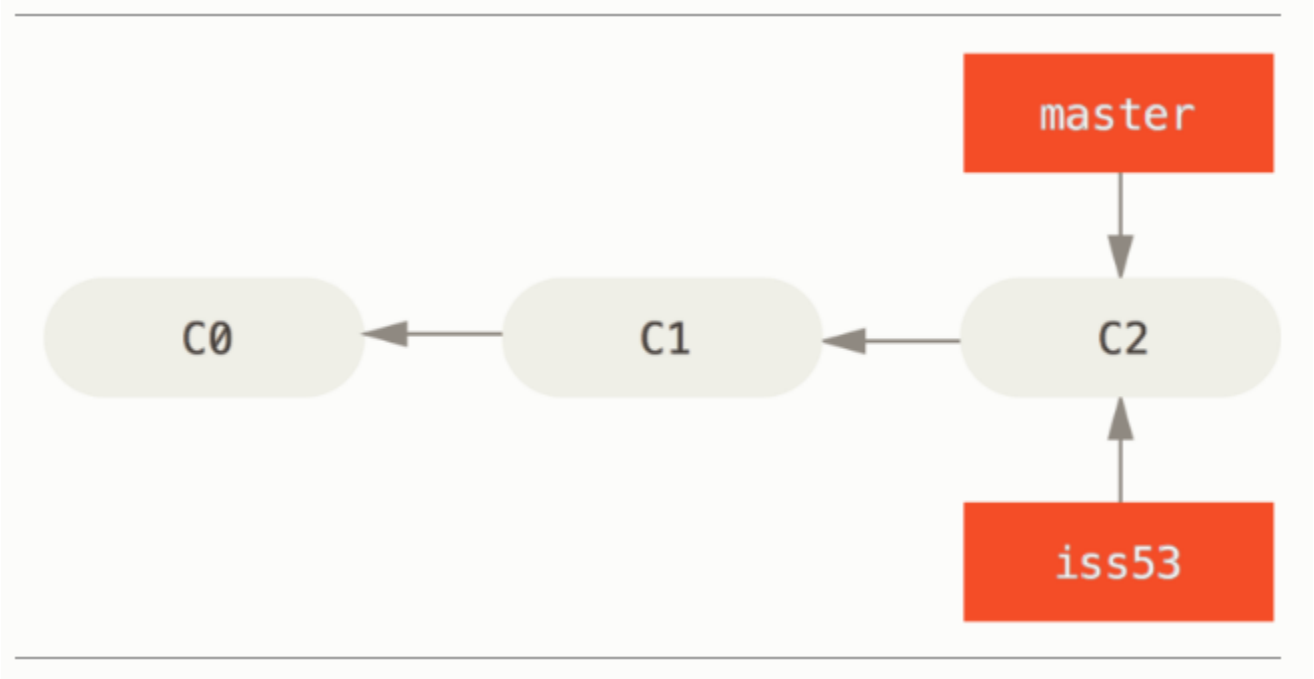
Branches are a part of your everyday development process.

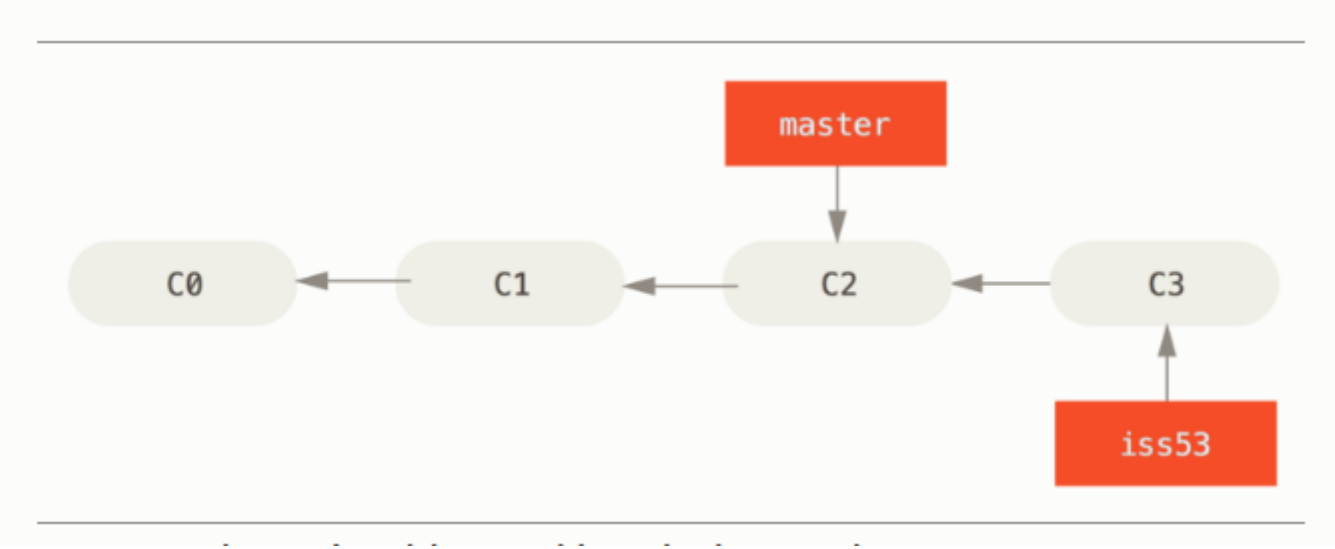
When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes.

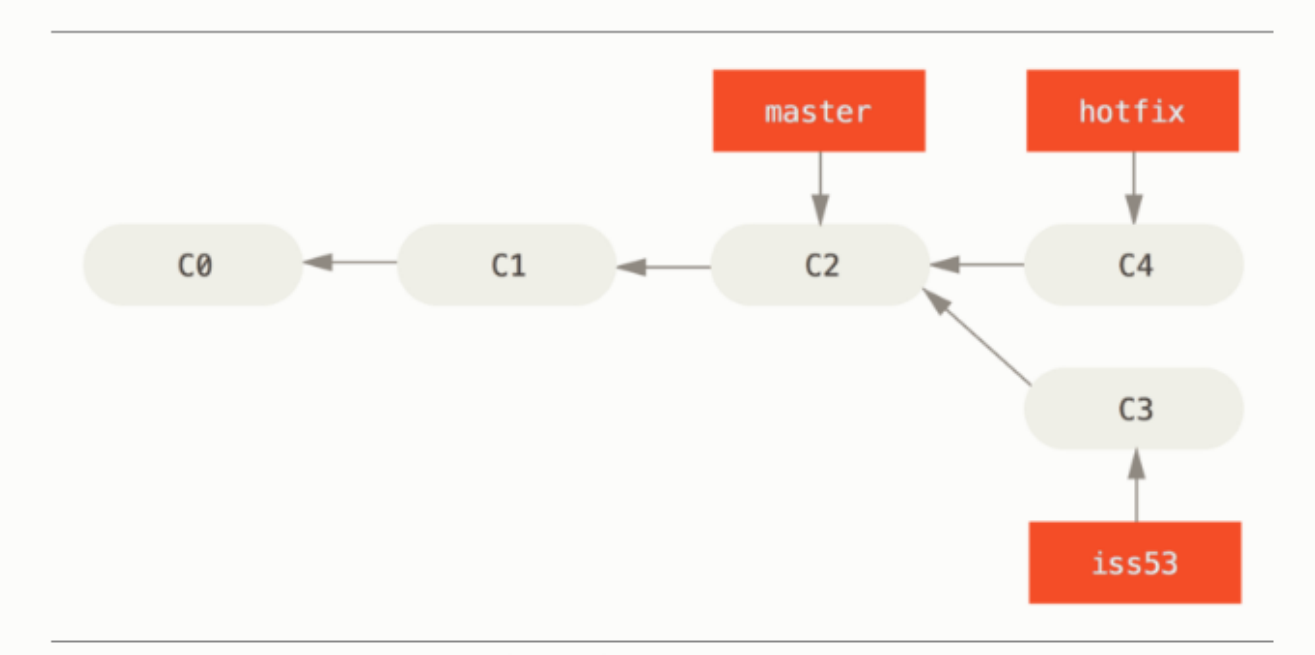
And allows your team to work on the same files in parallel!

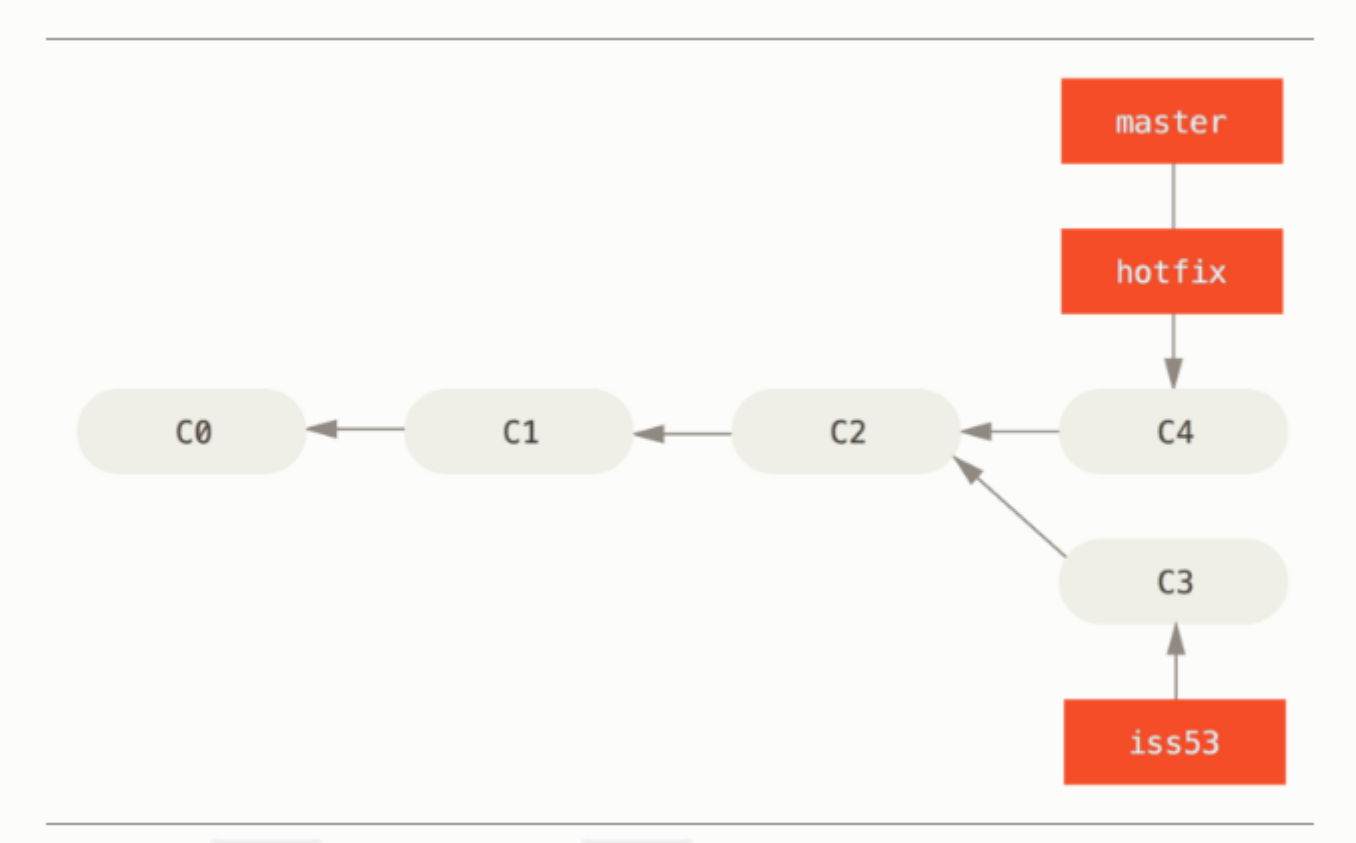
This makes sure that unstable code is never committed to the main code base, and it gives you the chance to clean up your feature's history before merging it into the main branch.

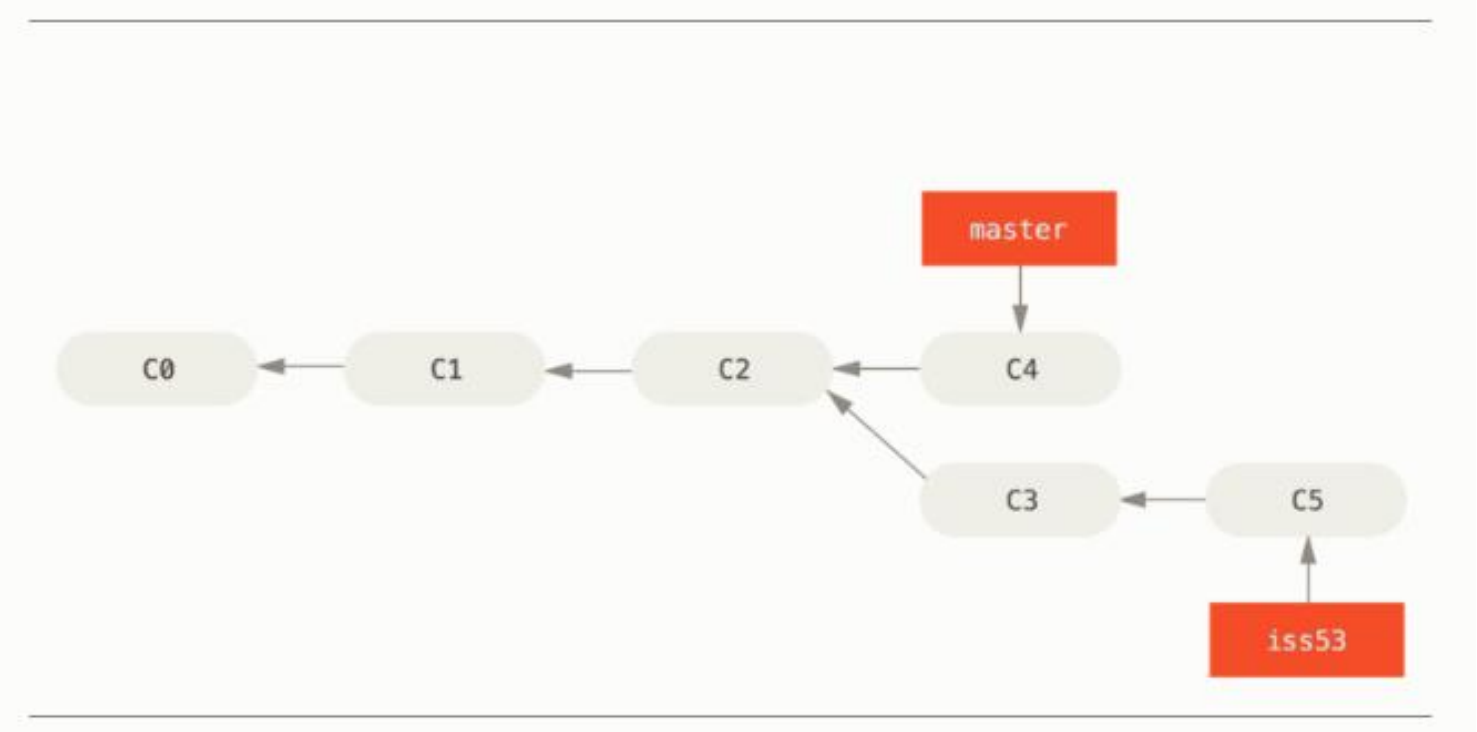

master
C0
C1
C2

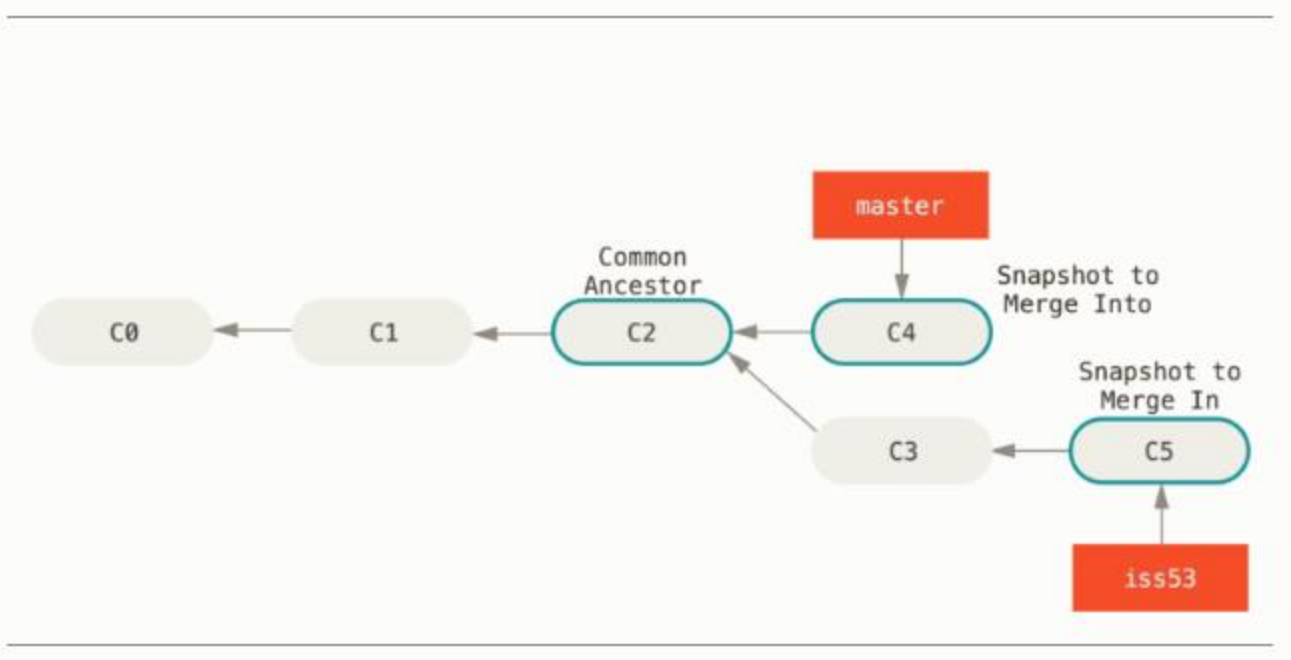

master
C0
C1
C2
iss53

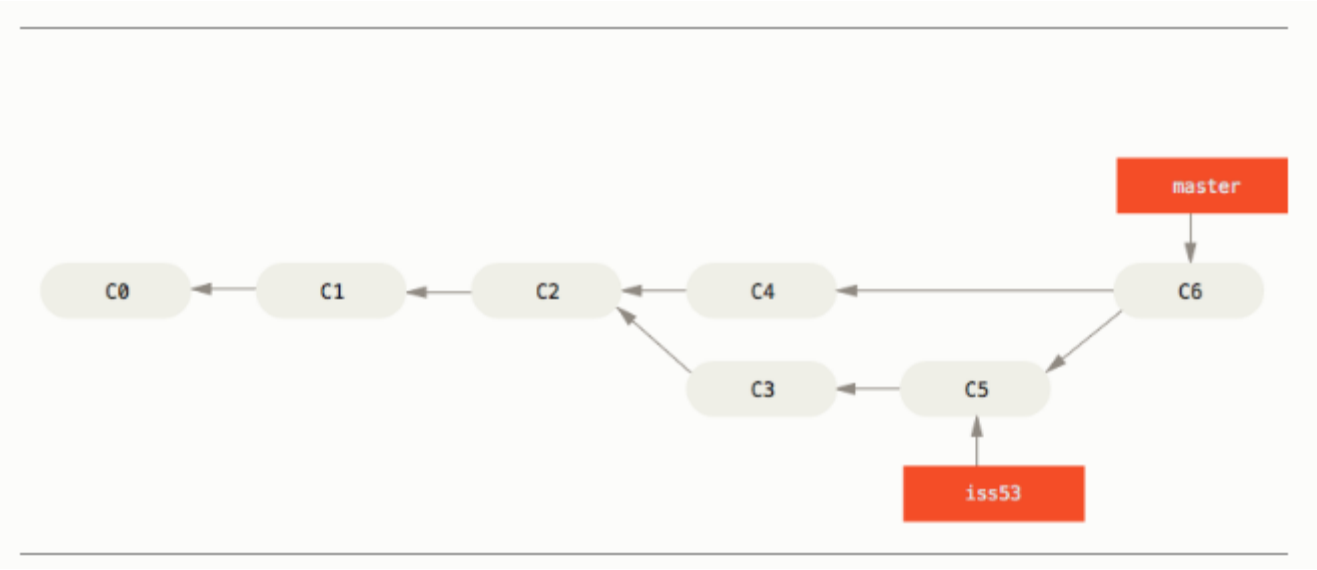

master
C0
C1
C2
C3
iss53


master
hotfix
C0
C1
C2
C4
C3
iss53

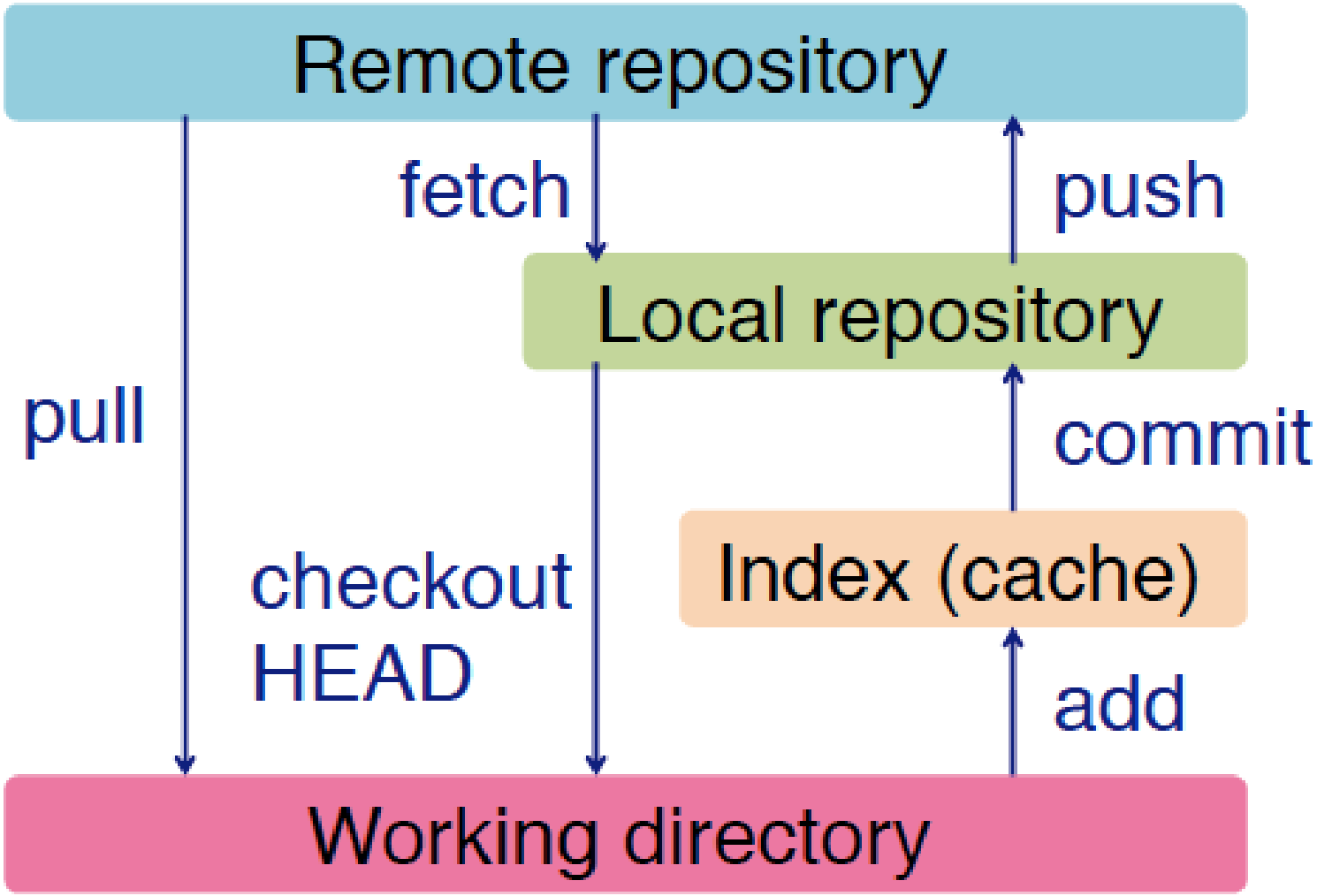

master
hotfix
C0
C1
C2
C4
C3
iss53


master
C0
C1
C2
C4
C3
C5
iss53


master
Common
Ancestor
Snapshot to
Merge Into
C0
C1
C2
C4
Snapshot to
Merge In
C3
C5
iss53


master
C0
C1
C2
C4
C6
C3
C5
iss53

Overview Git Workflow



Git Workflow Summary

Initial Setup

- Install git if not installed
  - **git config --global user.name "Your Name"**
  - **git config --global user.email yourname@colorado.edu**
  - git config --global color.ui true
- Create an account on GitHub
- Set up ssh https://help.github.com/articles/generating-ssh-keys/
- Access existing project on GitHub
  - **git remote add origin https://github.com/username/myproject.git**

Typical Workflow

- Write code
- Get ready, verify branch
  - **git status**
- If we created new files, add them
  - **git add filename**
- Commit changes
  - **git commit –a**
  - [add commit message in editor]
  - Or **git commit –m “msg”**
- Abandon all changes since last commit
  - **git reset –hard HEAD**

git tag ‘v0.0.1’ <commit id>
Commit id can be got by running the **git log** command

git push origin master
git push --tags

Branching

- Pull the master branch
  - **git checkout master**
  - **git pull**
- Branching. When starting a new task, we create a new branch. If we are continuing previous work, we simply switch to an existing branch.
- Creating a new branch. The command below both creates a new branch and switches to it.
  - **git checkout -b branchname**

    where branchname is the name of the branch we're creating.
- Switching to an existing branch
  - **git checkout branchname**

Typical Workflow

- Merging from master – at least once per day
  - **git checkout master**
  - **git pull**
  - **git checkout branchname**
  - **git merge master**
  - [if edit files to resolve conflicts then commit again]
  - git push origin branchname
- Merging into master
  - **git checkout master**
  - **git merge branchname**
  - **git push origin master**

Version Control – Best Practices

Do

+ Commit Early, Commit Often
+ Use Good Commit Messages
+ Play Nice with Others (e.g. fix your own merge conflicts)

Don’t

- Commit Generated/Compiled/Output Files (Source Only)
- Commit Secrets (e.g. credentials, passwords, etc)
- Merge Broken Code (i.e. don’t break the main branch)

Git Hosting

Git Hosting

GitHub: https://github.com/

- Git Only
- Unlimited Free Public Repos
- 5 Free Private Repos (with Student Discount)
- Unlimited Team Size
- Issue Tracking + Pull Requests/Reviews + Wiki + Websites

BitBucket: https://bitbucket.org/

- Git or Hg
- Unlimited Free Public Repos
- Unlimited Free Private Repos
- Unlimited Team Size (with Student Discount)
- Issue Tracking + Pull Requests/Reviews + Code Reviews

Git

CU CS GitLab: https://git.cs.colorado.edu/

- Git Only
- Unlimited Free Public Repos
- Unlimited Free Private Repos
- Unlimited Team Size
- Issue Tracking + Pull Requests/Reviews + Wiki
- Free and Open Source