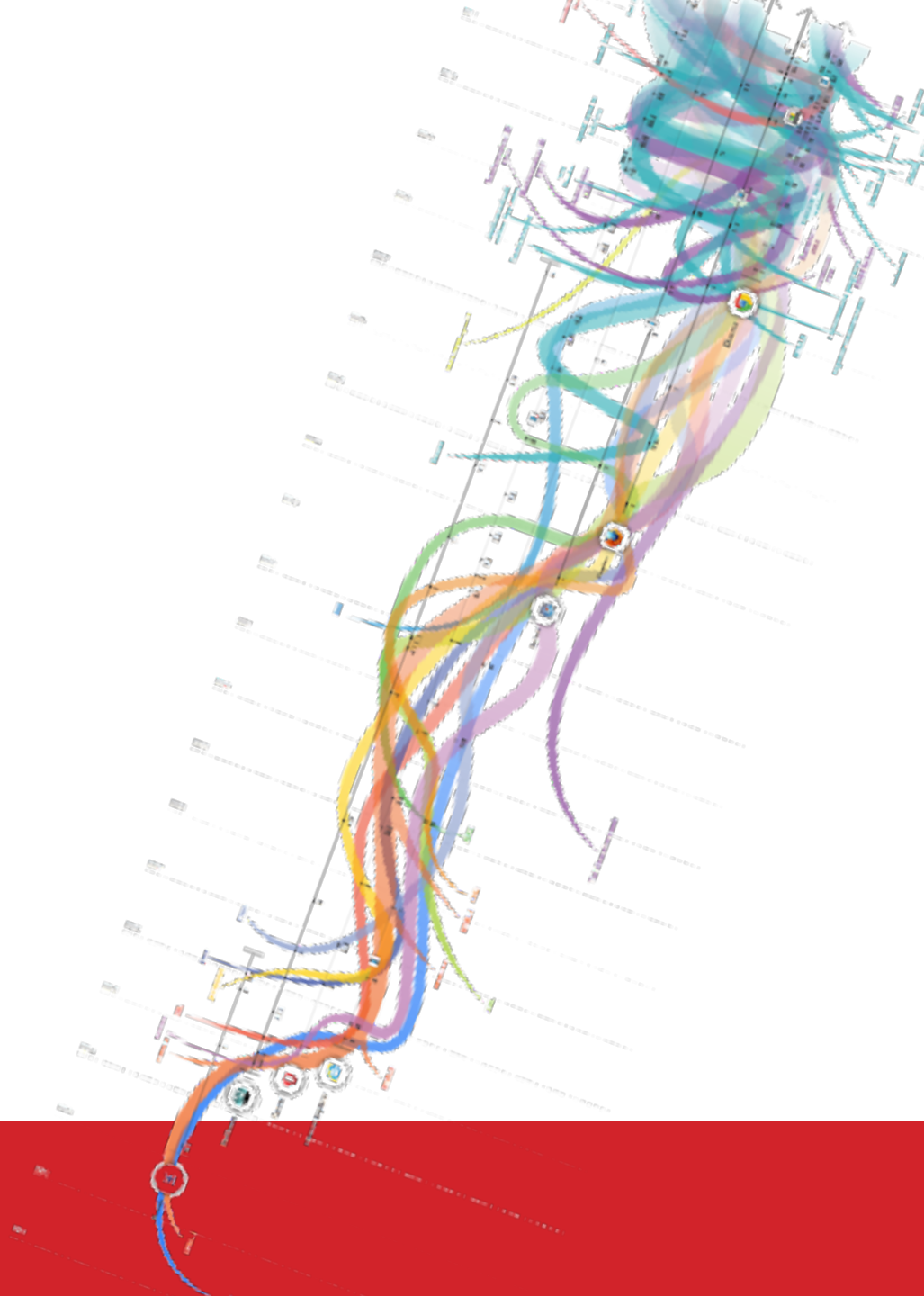
GitHUB

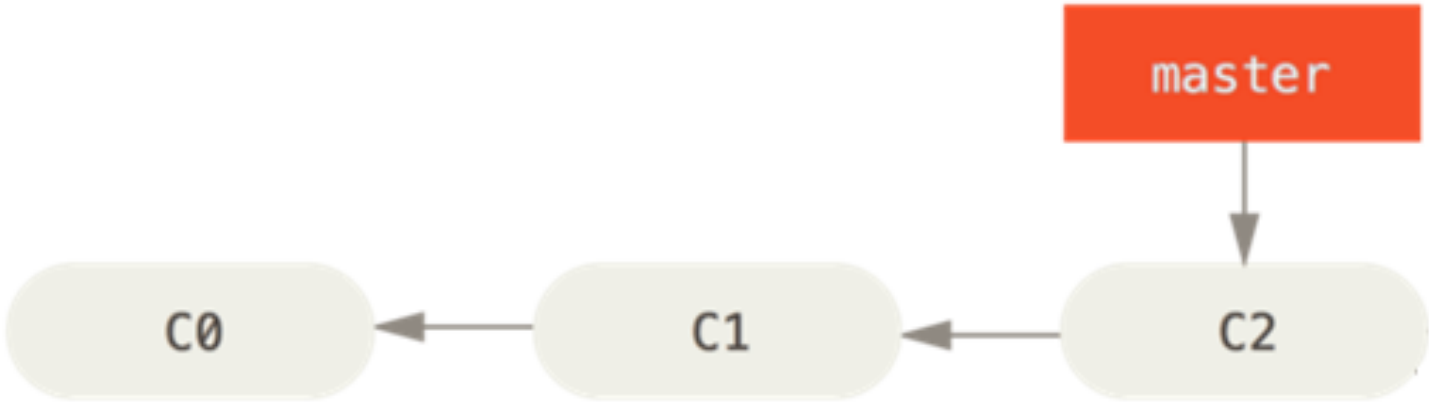
Metodologie di Programmazione per il Web



Why track/manage revisions?

Goal

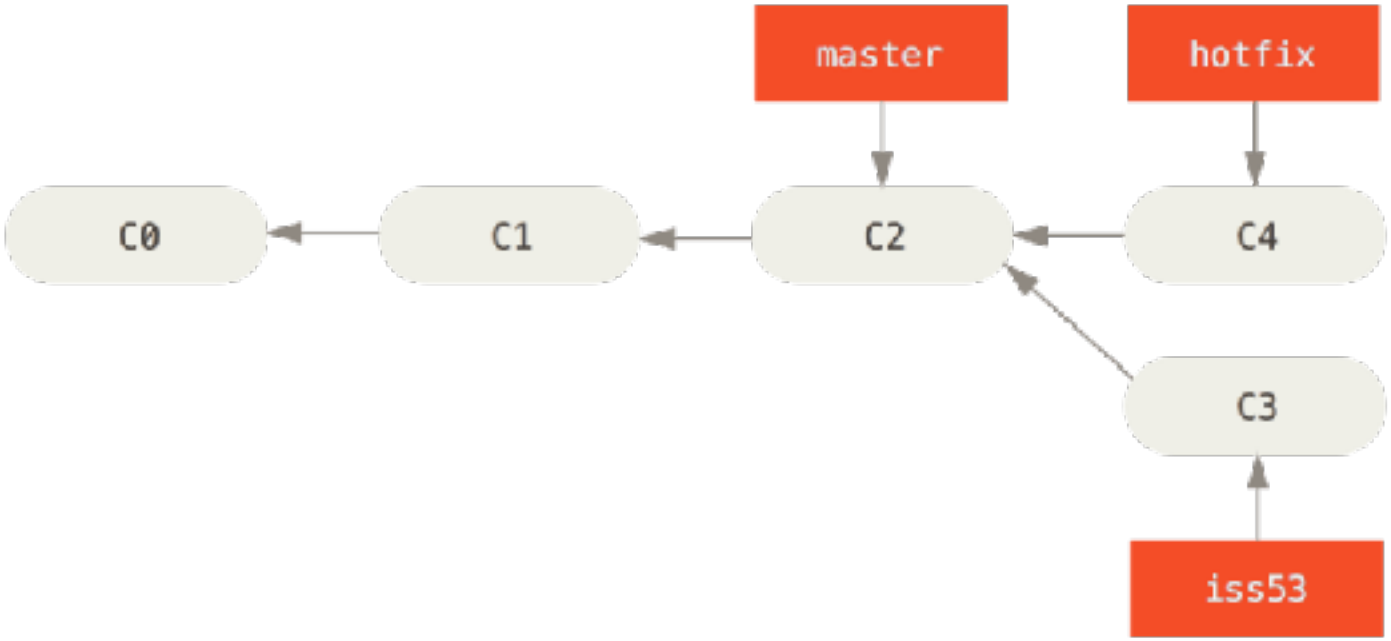
- **Backup**: Undo or refer to old stuff



http://git-scm.com/book/en/Git-Branching-Basic-Branching-and-Merging

Goal

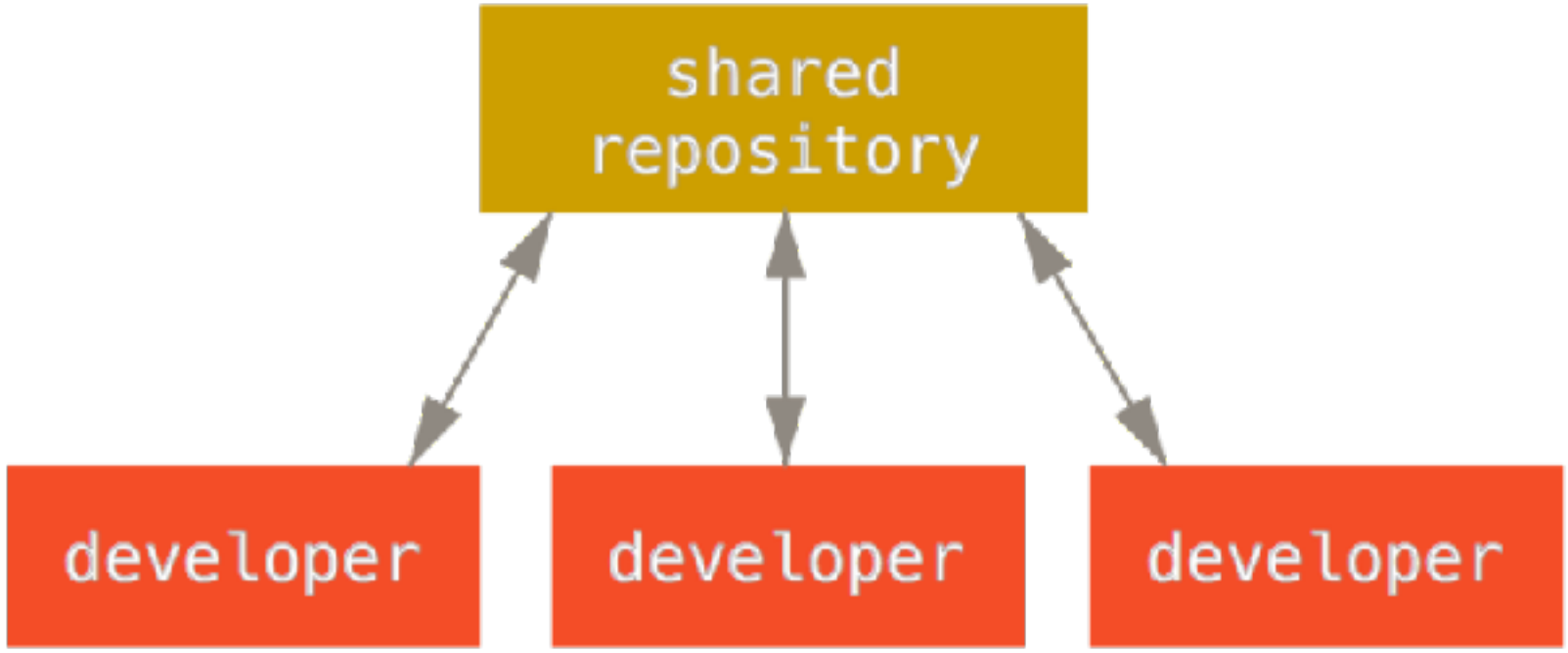
- **Branch**: Maintain old release while working on new



http://git-scm.com/book/en/Git-Branching-Basic-Branching-and-Merging

Goal

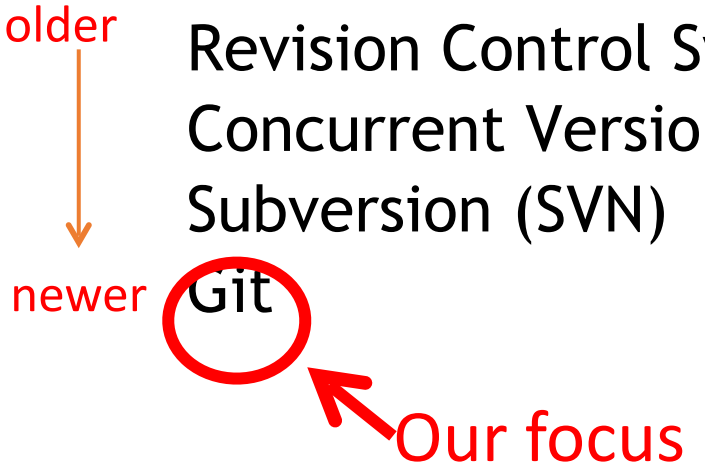
- **Collaborate**: Work in parallel with teammates

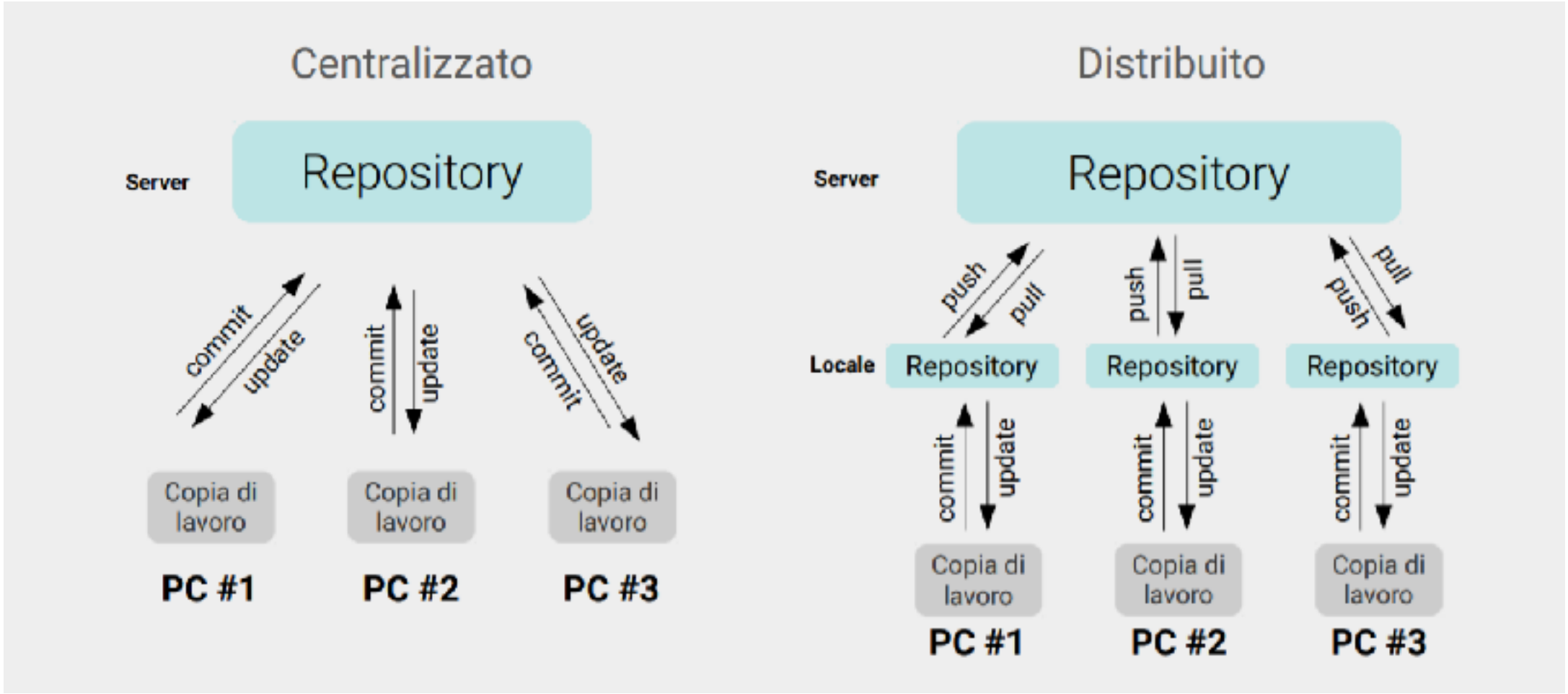


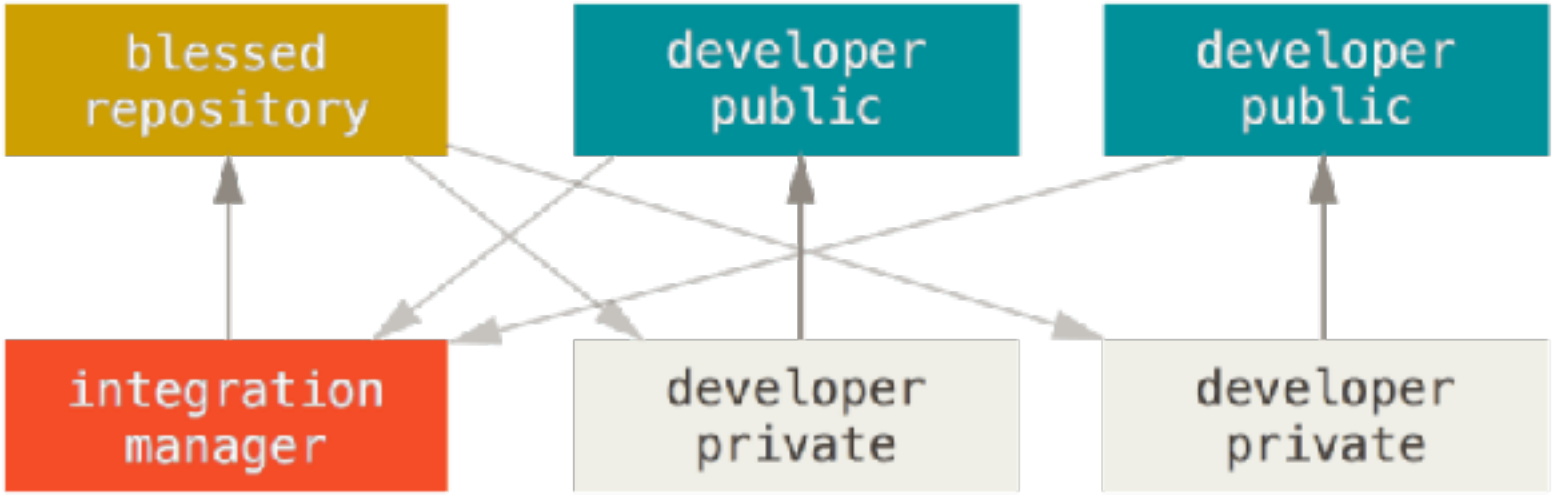
http://git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows

Version Control Systems (VCSs)

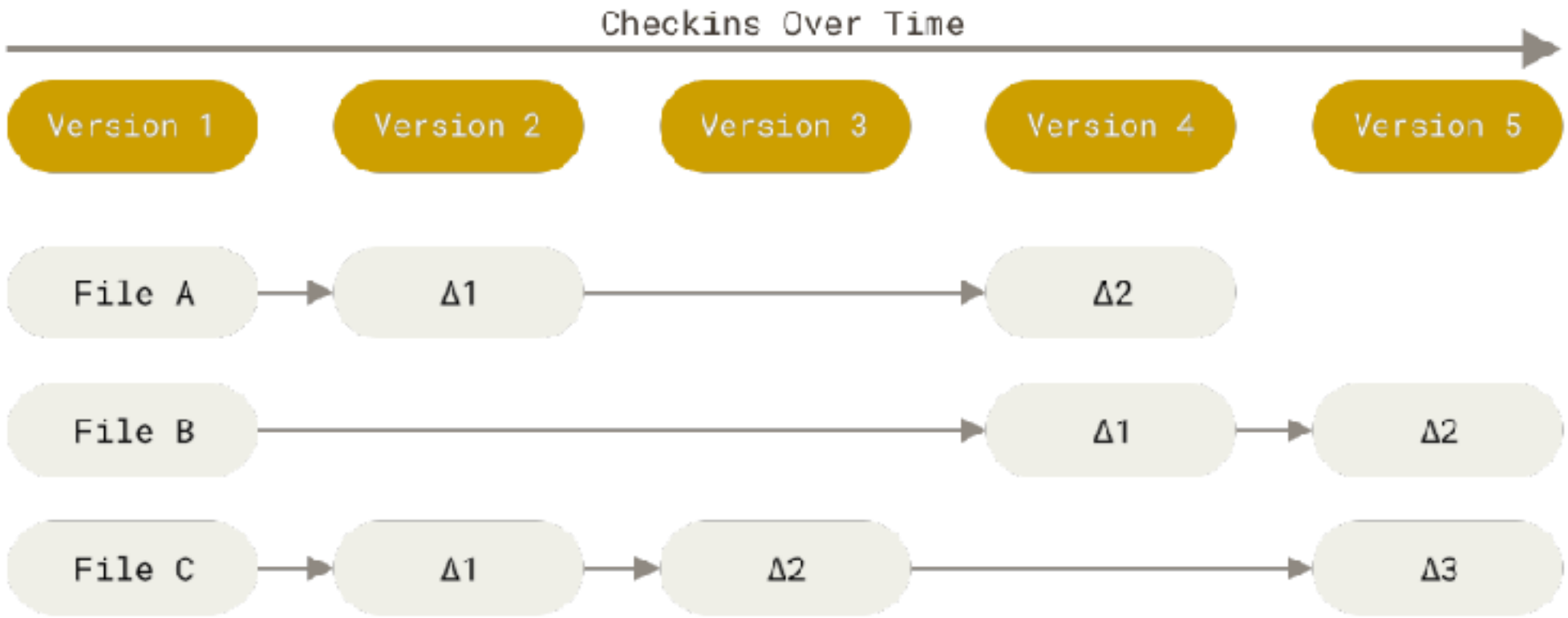
- Help you track/manage/distribute revisions
- Standard in modern development
- Examples:




Centralizzato
Server
Repository
commit
update
commit
update
update
commit
Copia di lavoro
Copia di lavoro
Copia di lavoro
PC #1
PC #2
PC #3
Distribuito
Server
Repository
push
pull
push
pull
pull
push
Locale
Repository
Repository
Repository
commit
update
commit
update
commit
update
Copia di lavoro
Copia di lavoro
Copia di lavoro
PC #1
PC #2
PC #3


blessed repository
developer public
developer public
integration manager
developer private
developer private

Snapshots, Not Differences



Configure your Git client

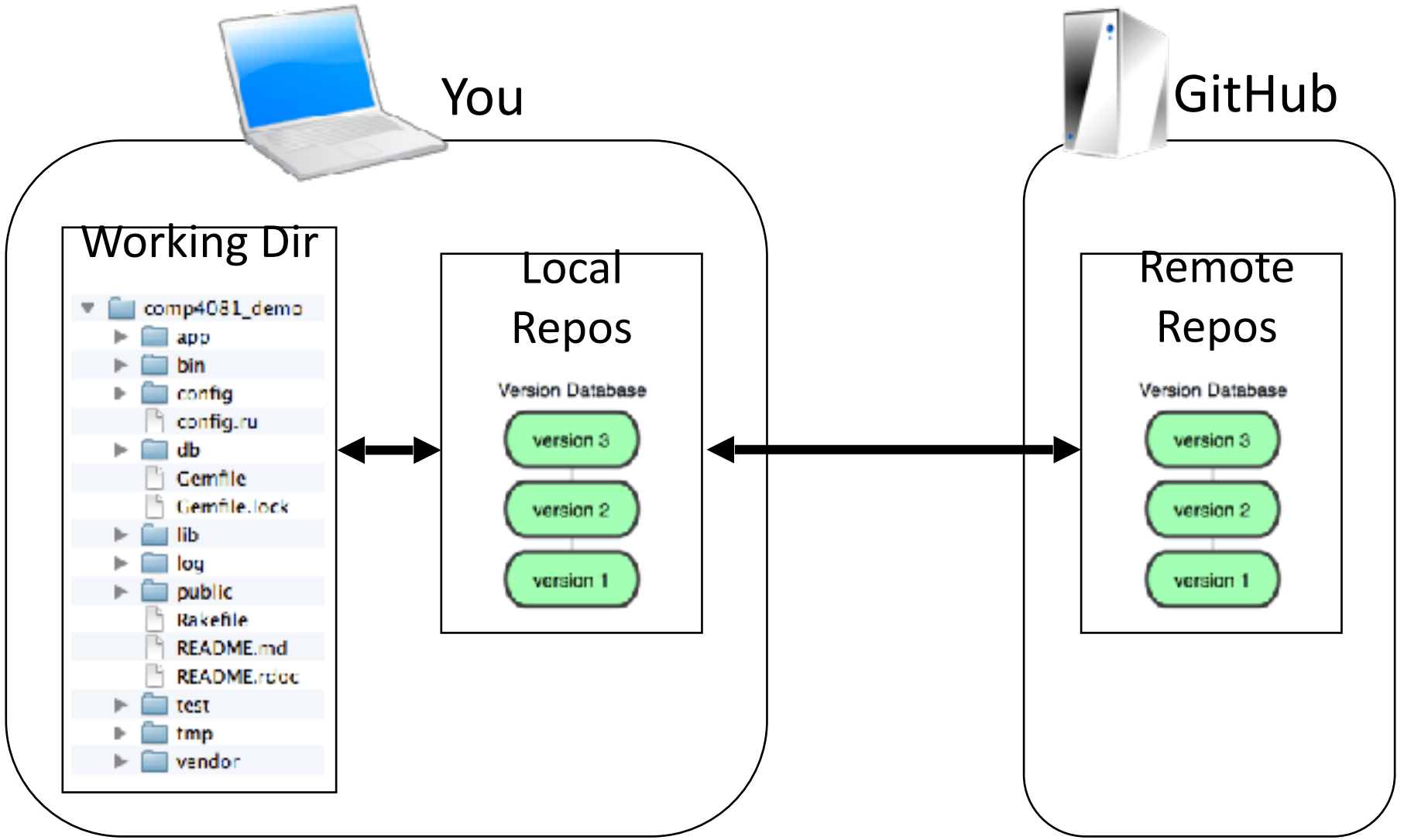
- Install Git
- Check config info:

```
$ git config --list --show-origin

user.name=Alessio Bottrighi
user.email=alessio.bottrighi@uniupo.it
```

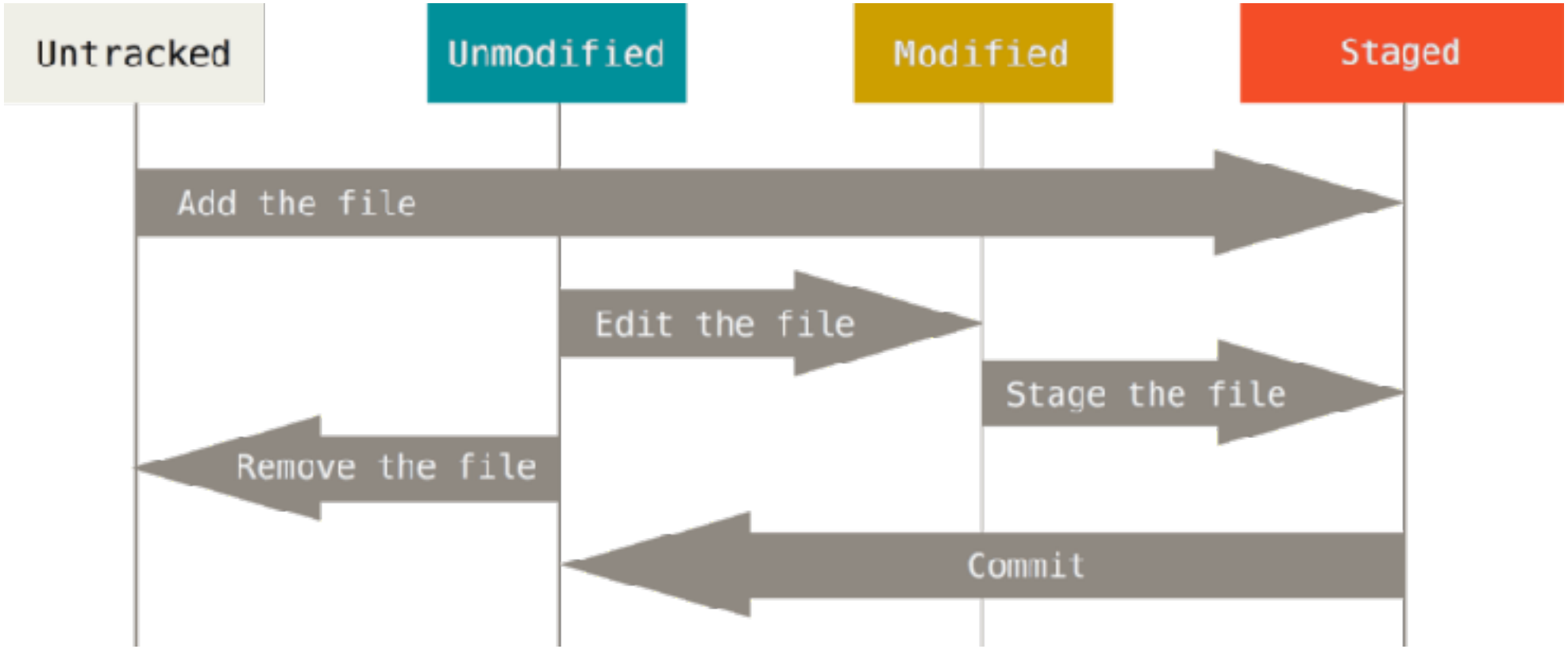
- Fix if necessary:

```
$ git config --global user.name "John Doe"

$ git config --global user.email jdoe@example.org
```

GitHub-User Perspective



The lifecycle of the status of your files

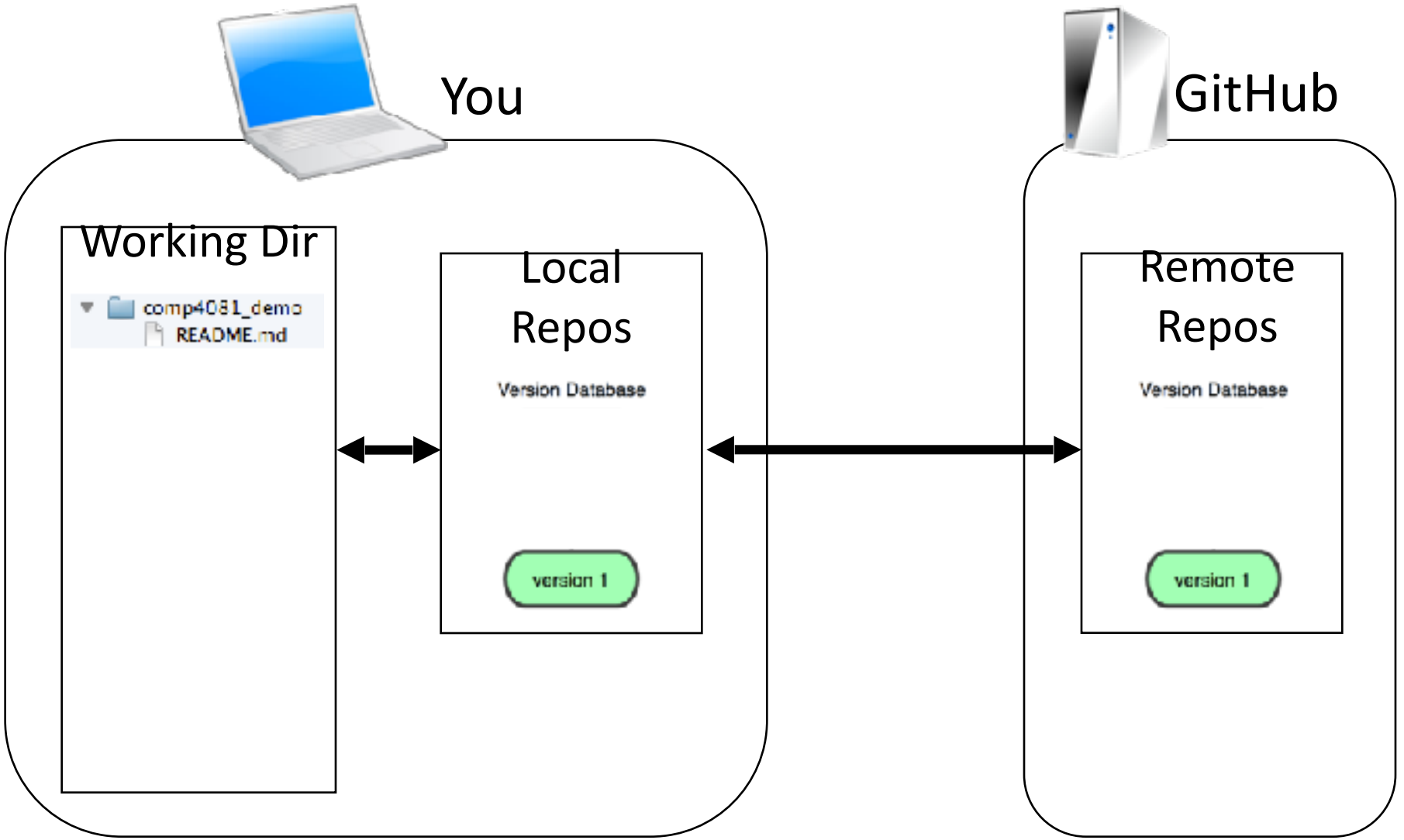


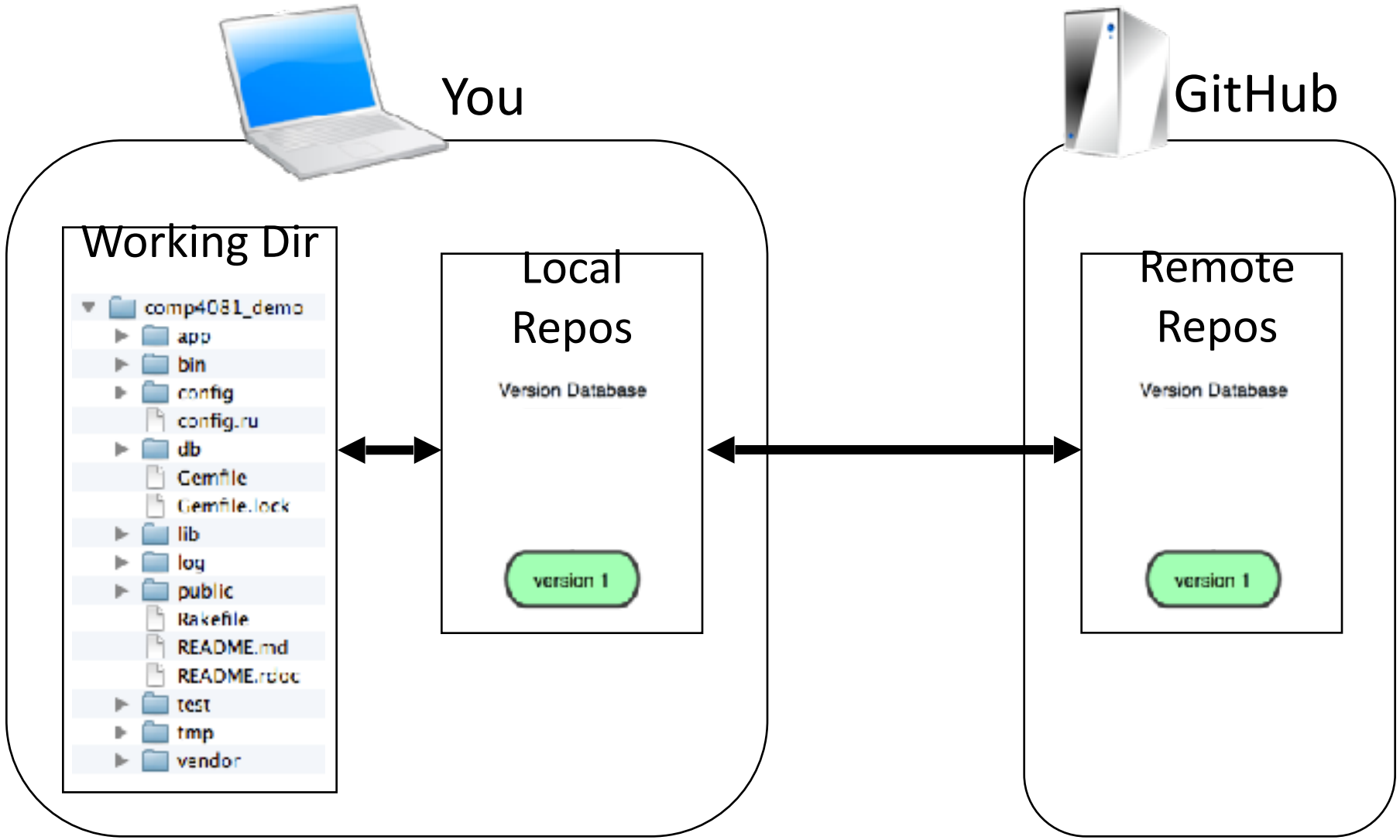
Let’s begin with an example…

You

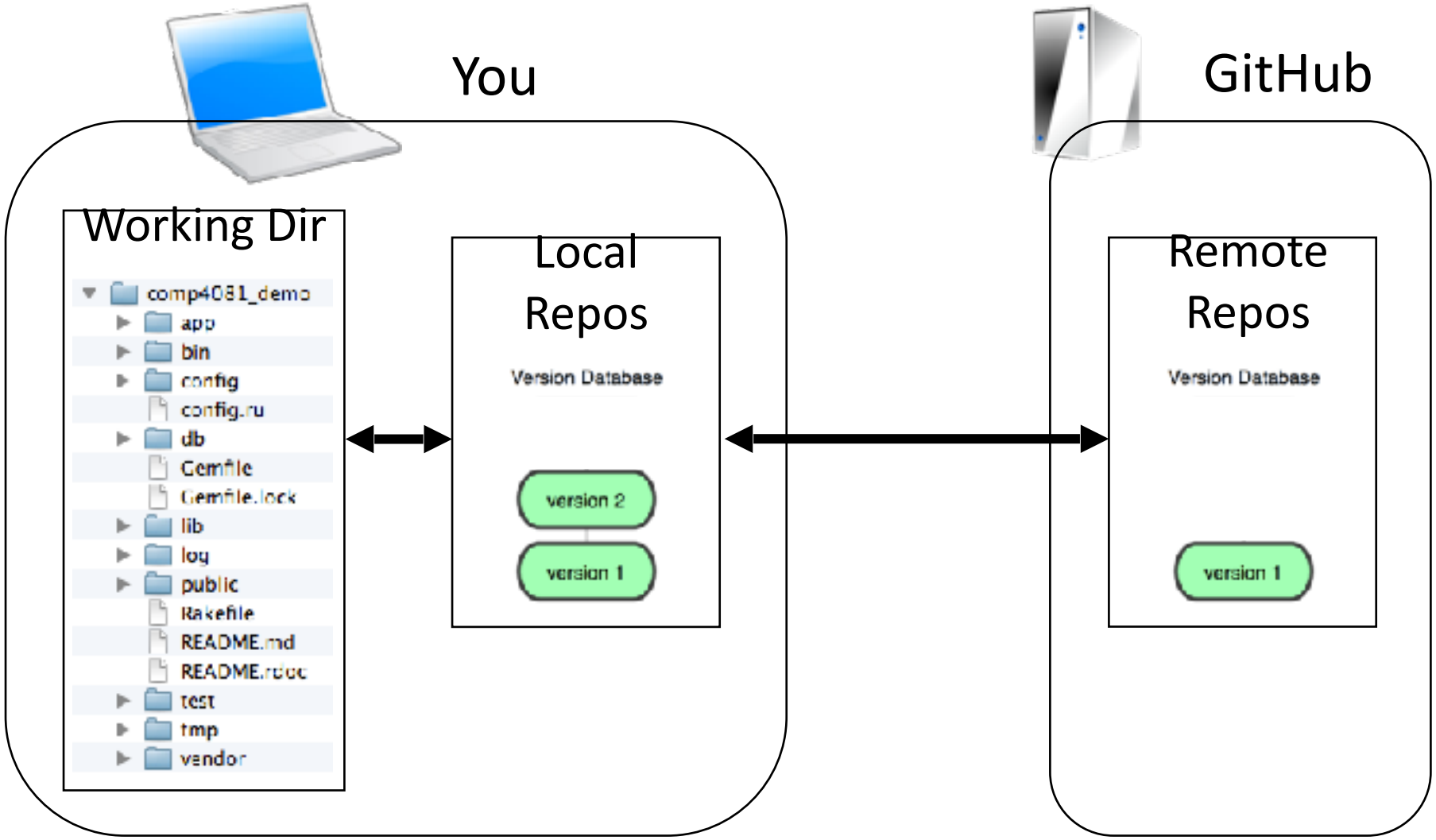
GitHub

```
$ git clone https://github.com/libgit2/libgit2
```

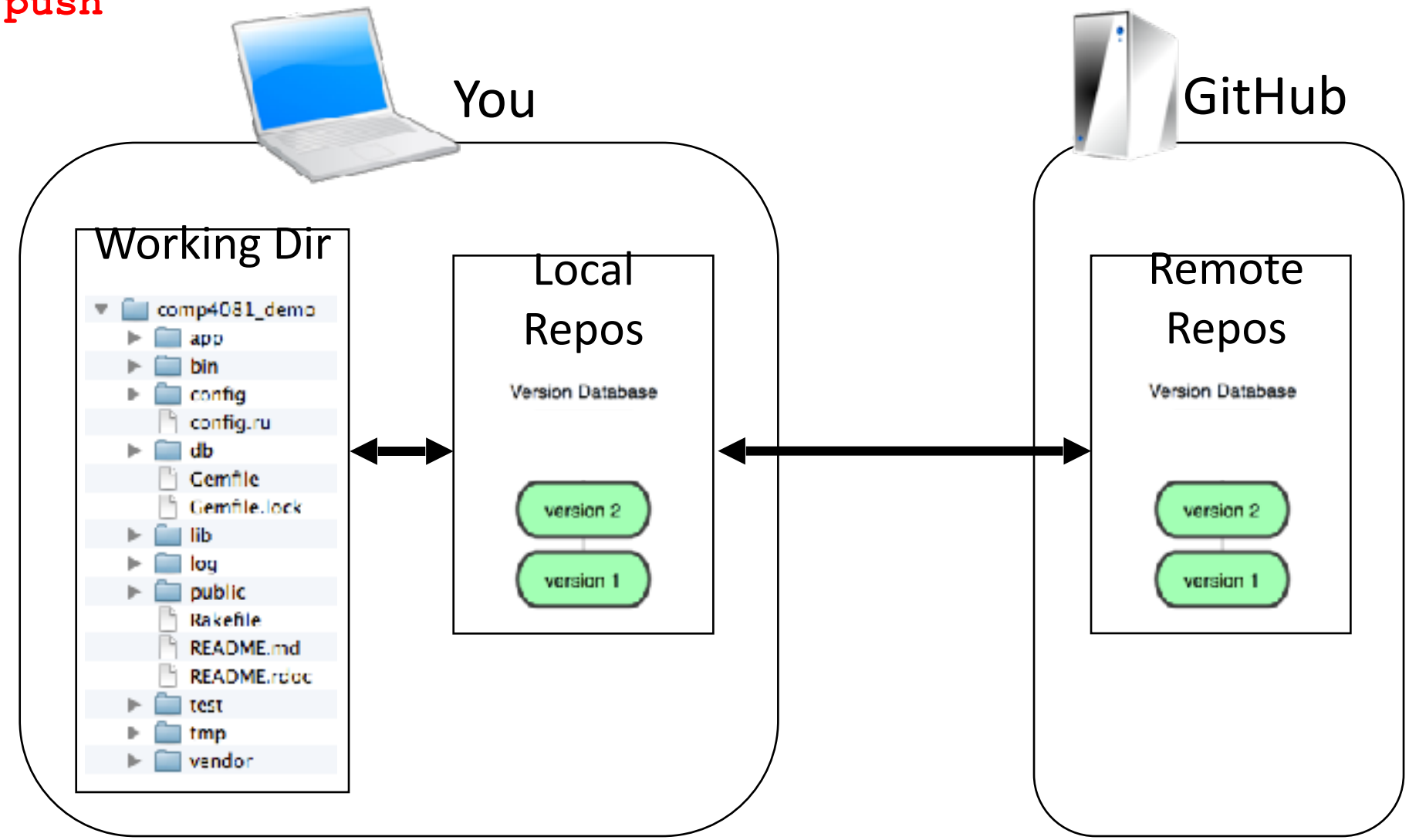



You
GitHub
Working Dir
comp4081_demo
app
bin
config
config.ru
db
Gemfile
Gemfile.lock
lib
log
public
Rakefile
README.md
README.rdoc
test
tmp
vendor
Local
Repos
Version Database
version 1
Remote
Repos
Version Database
version 1

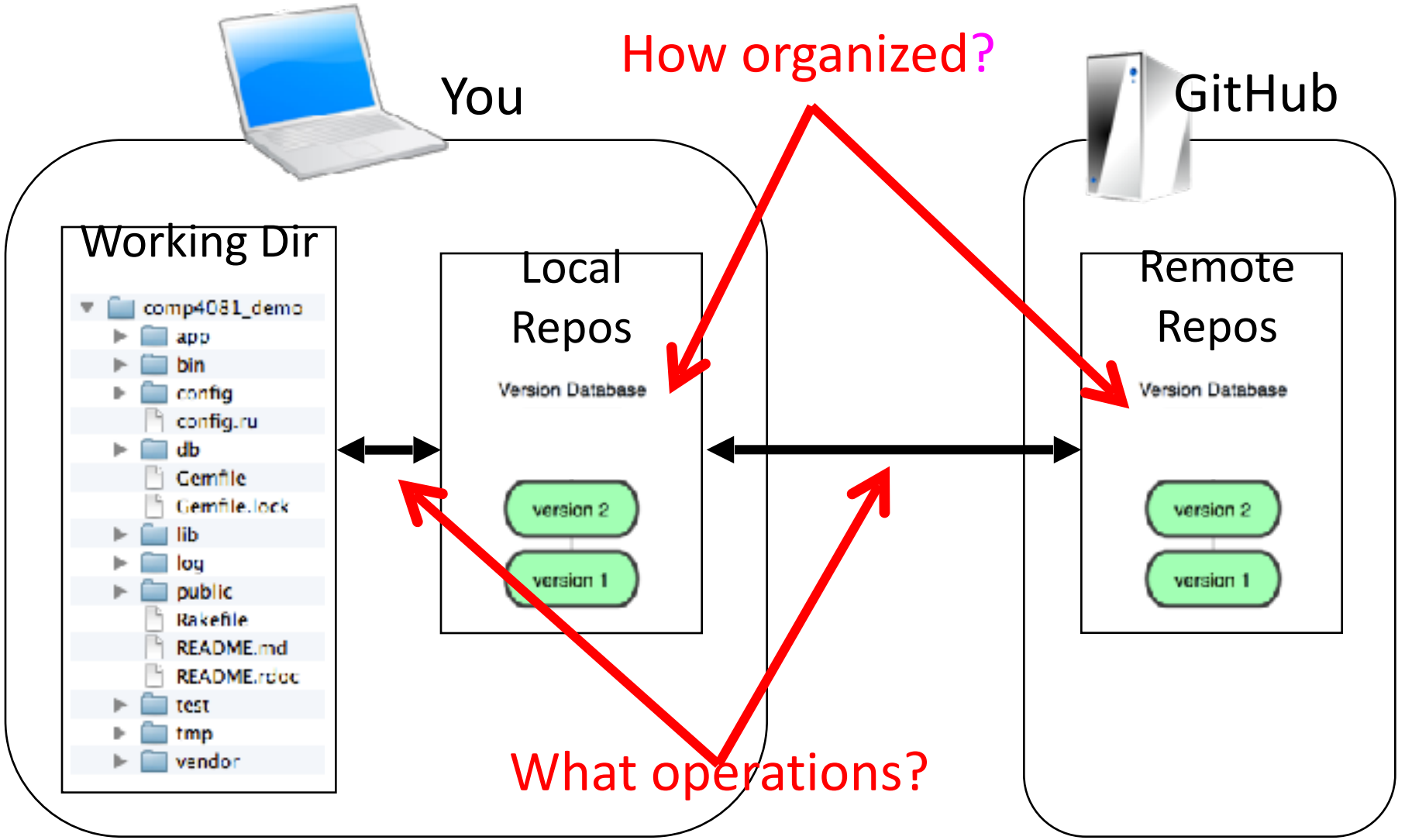
```
$ cd comp4081_demo
$ git add -A
$ git commit -m "Created project skeleton"
```



```
$ git push
```

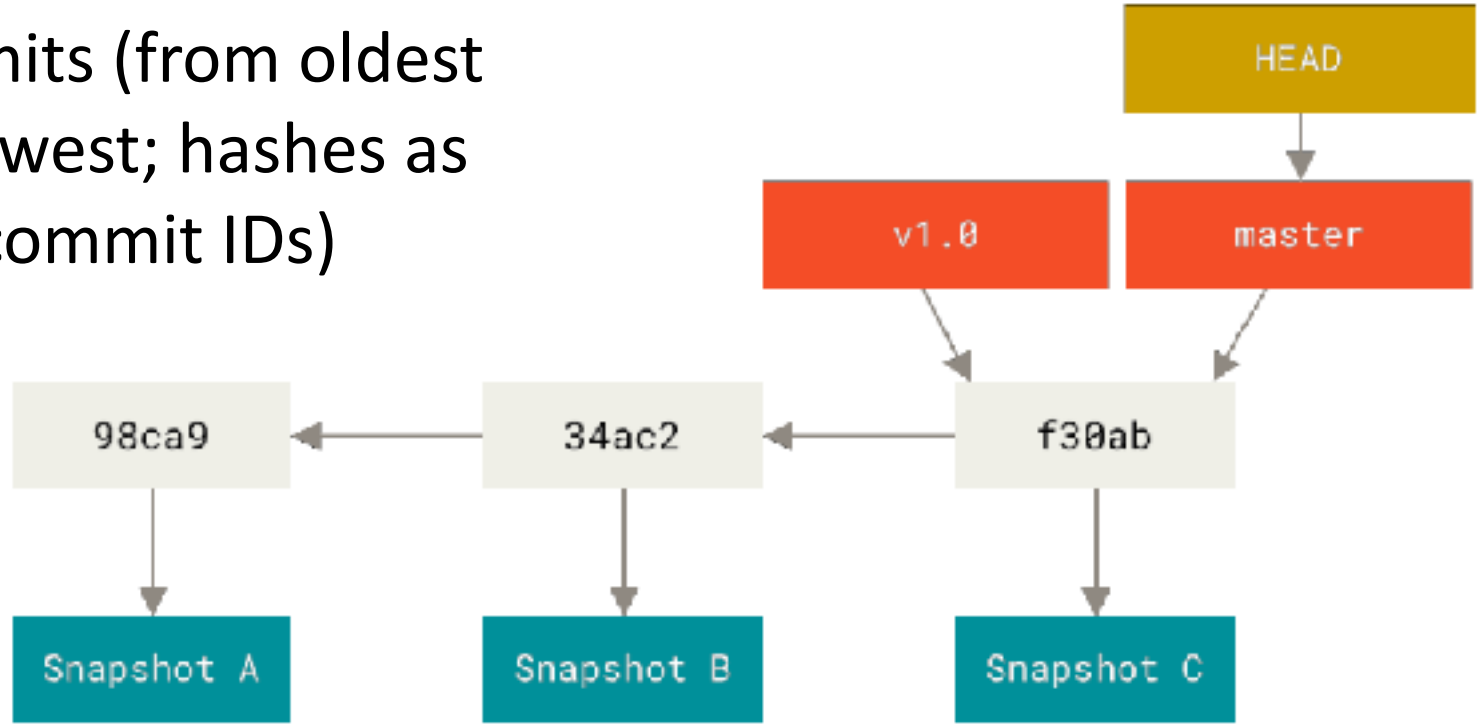

You
GitHub
Working Dir
comp4081_demo
app
bin
config
config.ru
db
Gemfile
Gemfile.lock
lib
log
public
Rakefile
README.md
README.rdoc
test
tmp
vendor
Local
Repos
Version Database
version 2
version 1
Remote
Repos
Version Database
version 2
version 1

Questions to answer

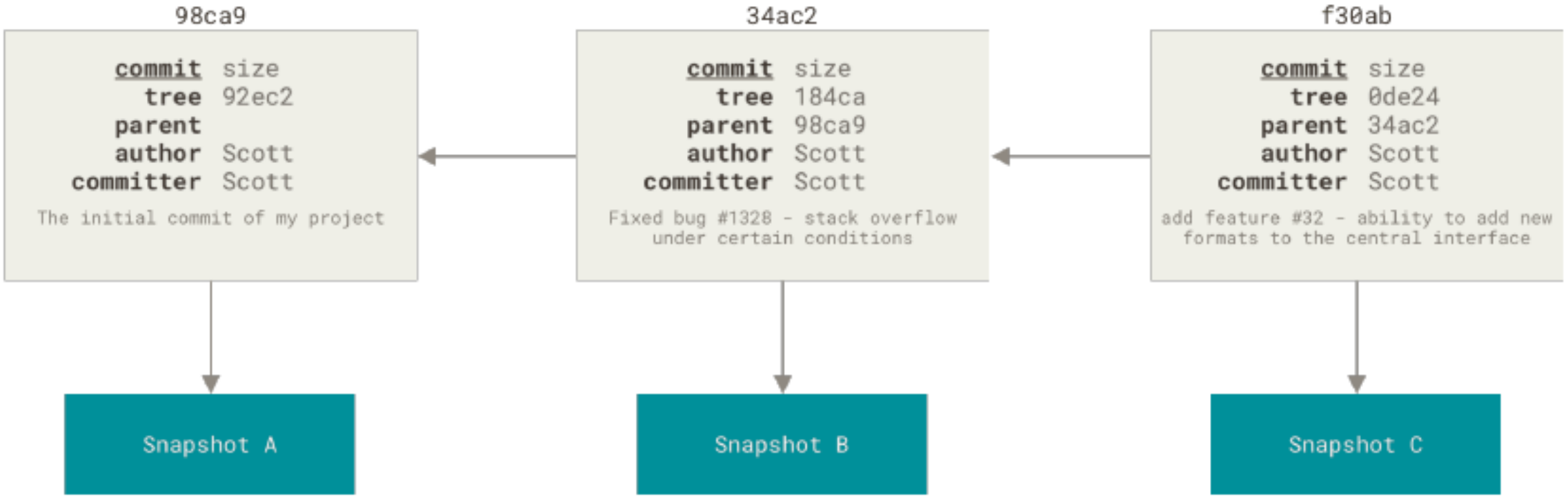


How the repos is organized

Commits (from oldest to newest; hashes as commit IDs)

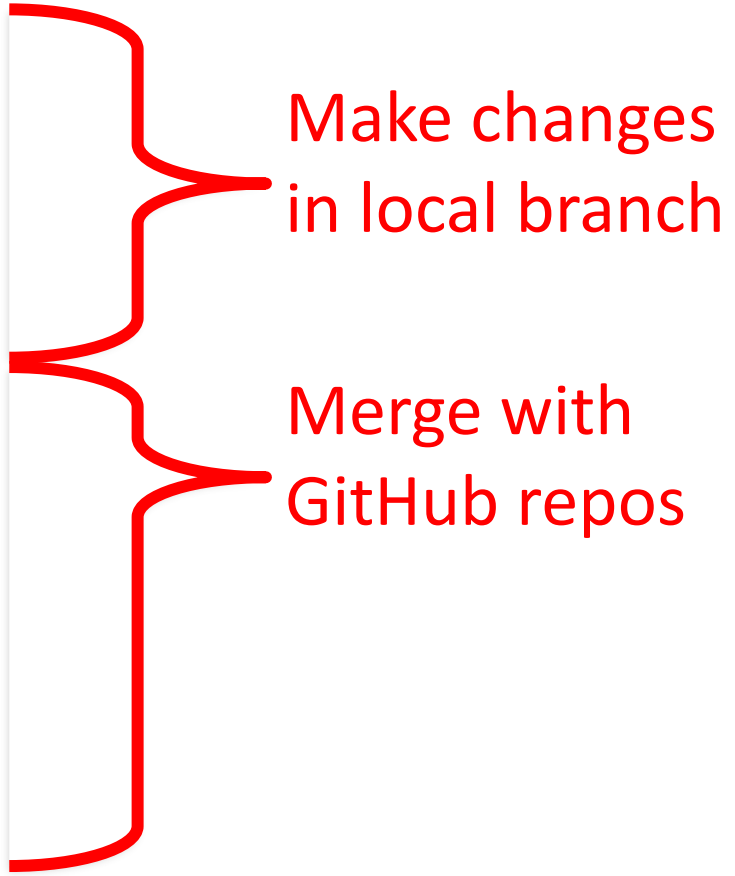


How the repos is organized



Common Workflow

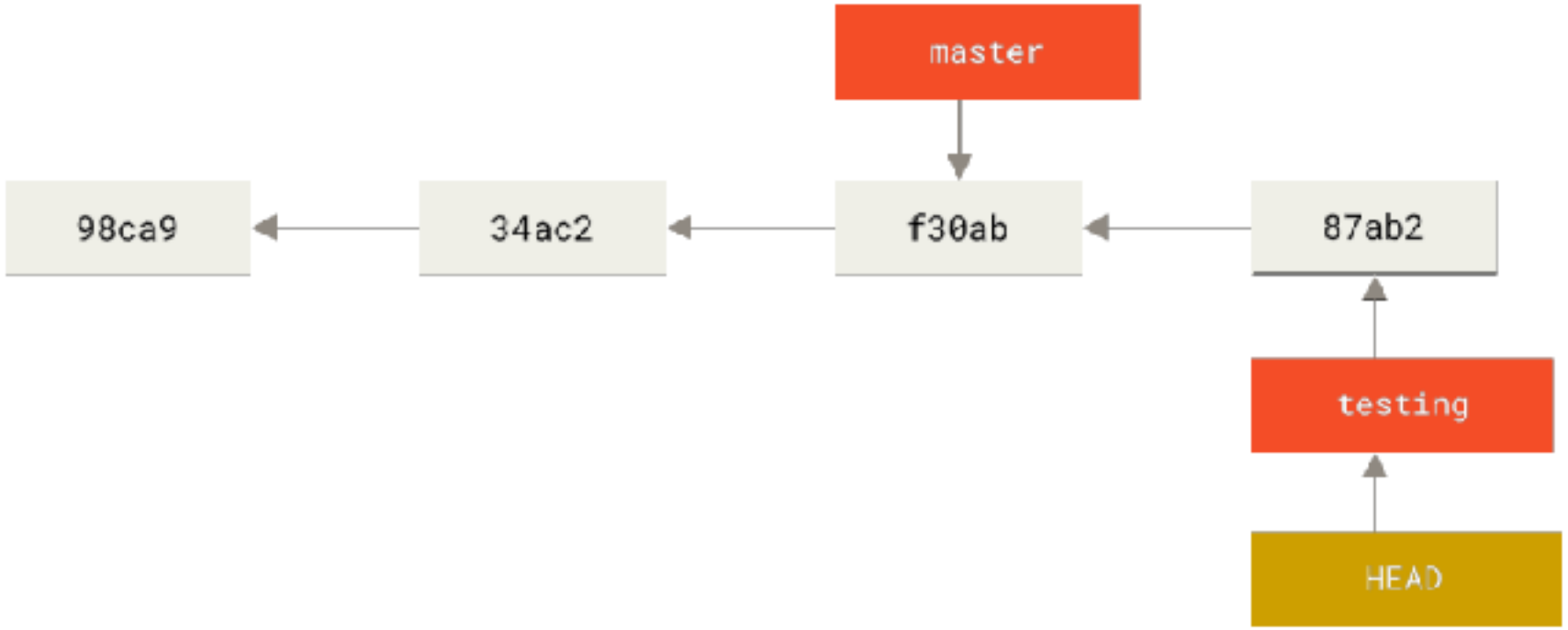
1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos



Organization with two branches

$ git branch testing

Organization with two branches

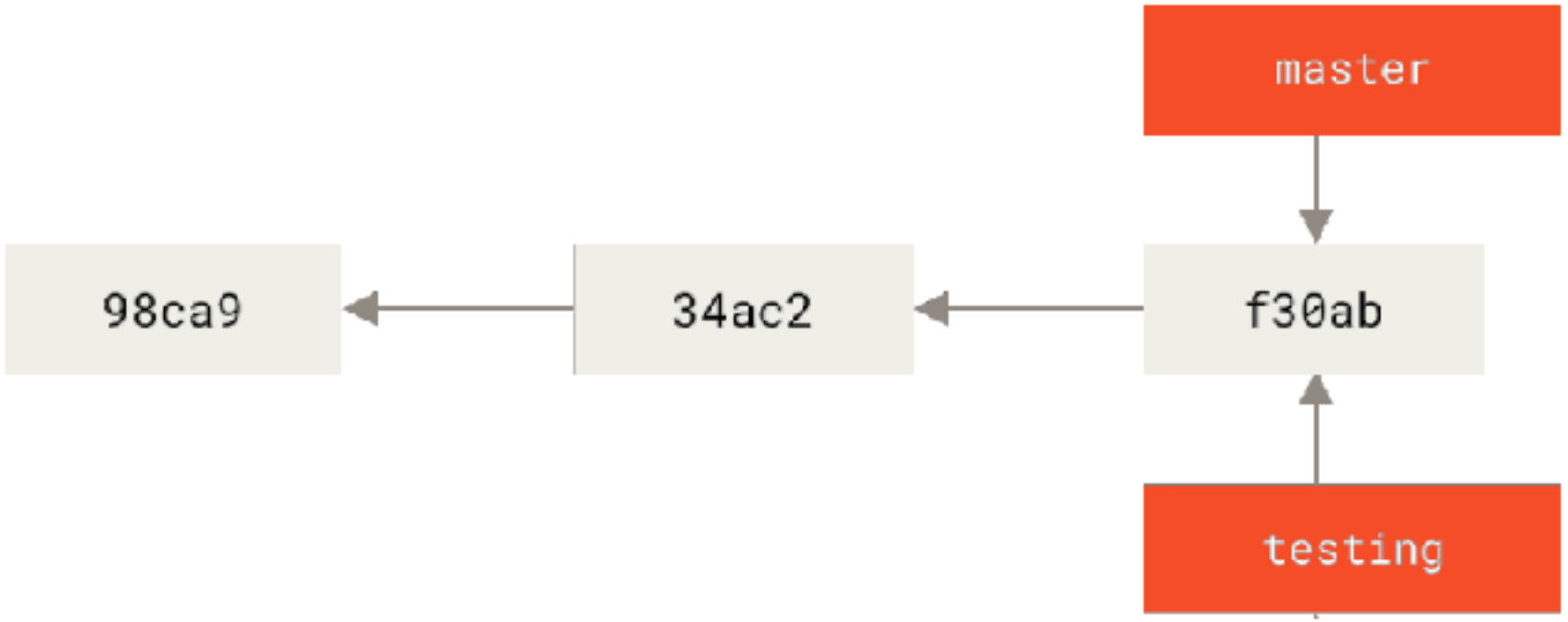


Common Workflow

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git branch works

```
$ git branch testing
```

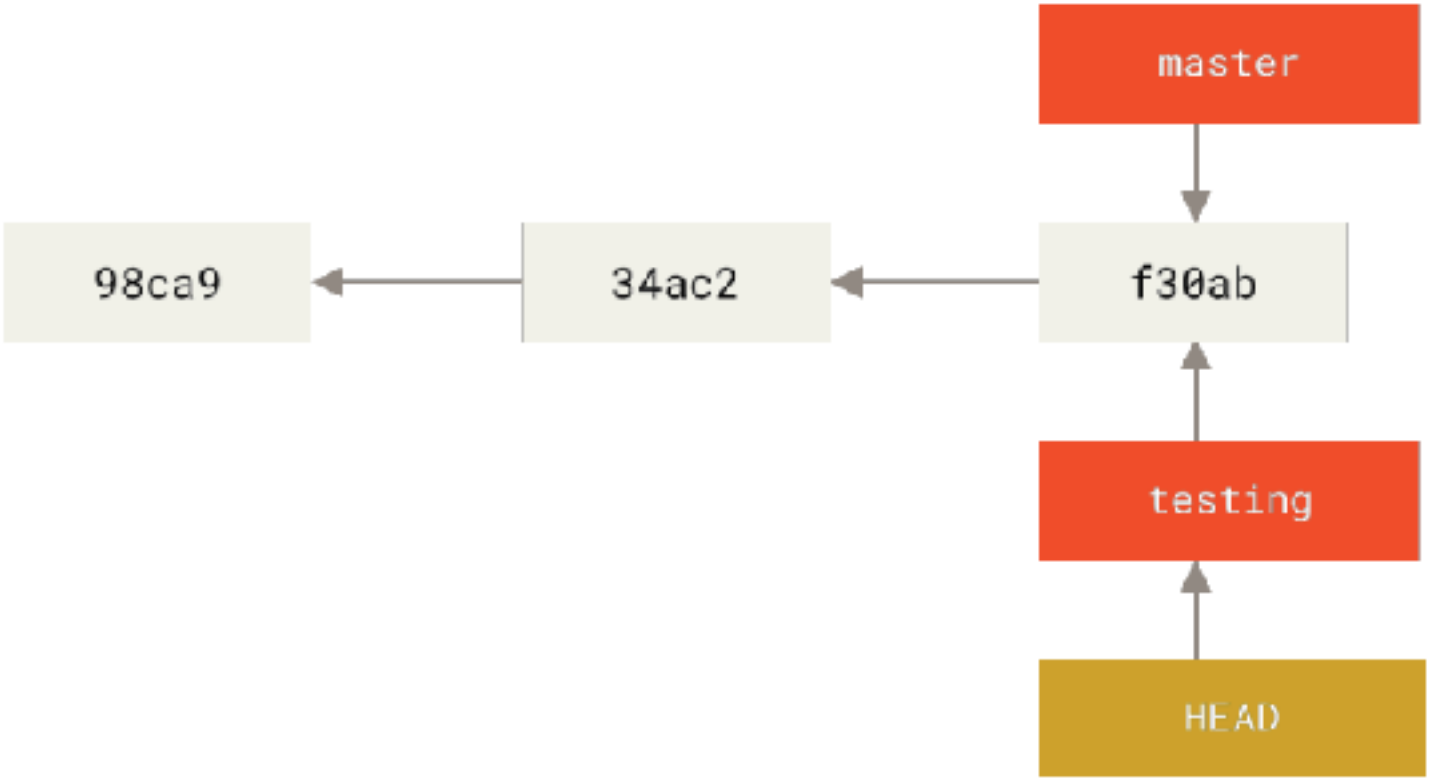


Common Workflow

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git checkout works

```
$ git checkout testing
```

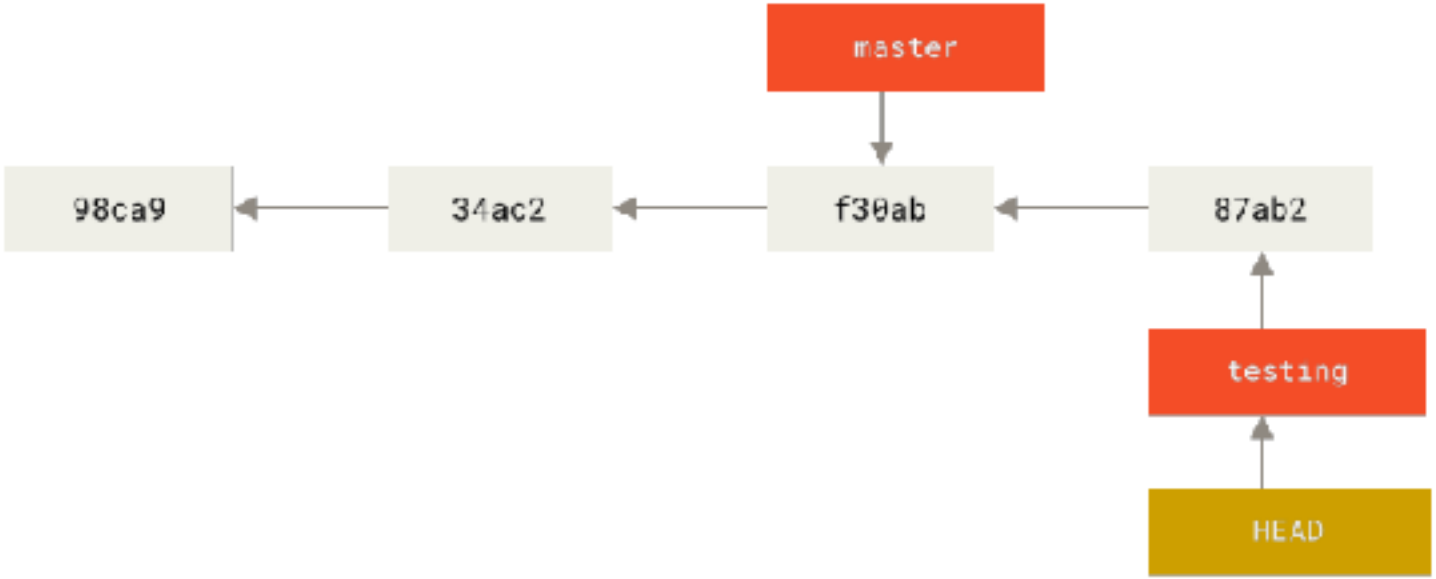


Common Workflow

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git commit works with multiple branches

```
Edit some stuff
$ git add -A
$ git commit -m "blah"
```



Common Workflow

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git checkout works

```
$ git checkout master
```

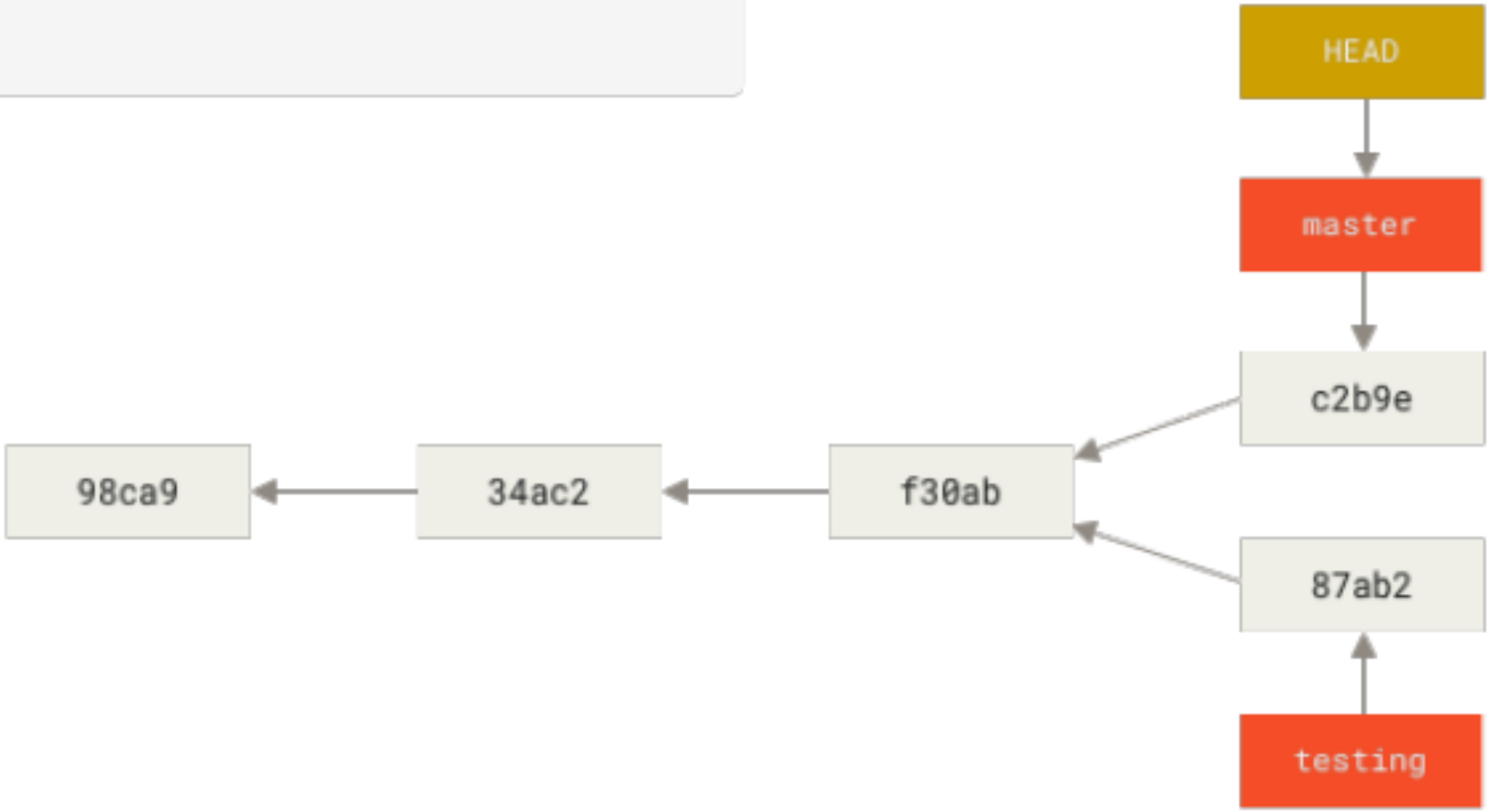
Common Workflow

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How git pull works

How git pull works

```
$ vim test.rb
$ git commit -a -m 'made other changes'
```



How git pull works

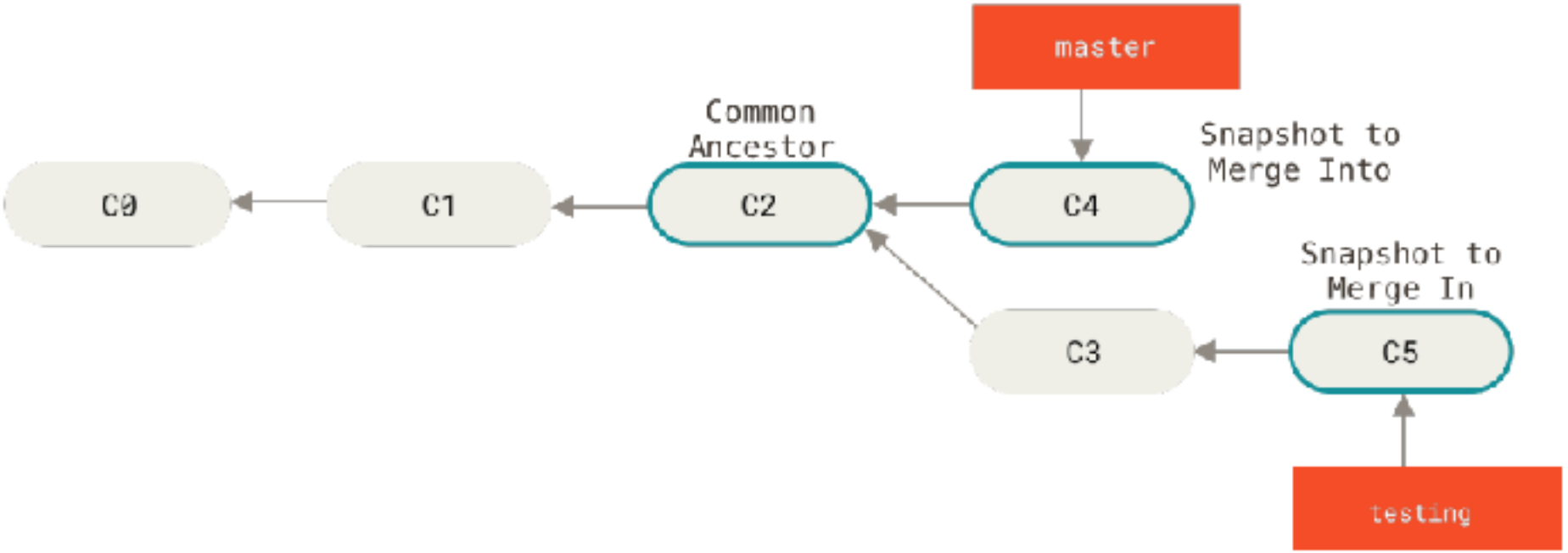
```
$ git pull
```

Common Workflow

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

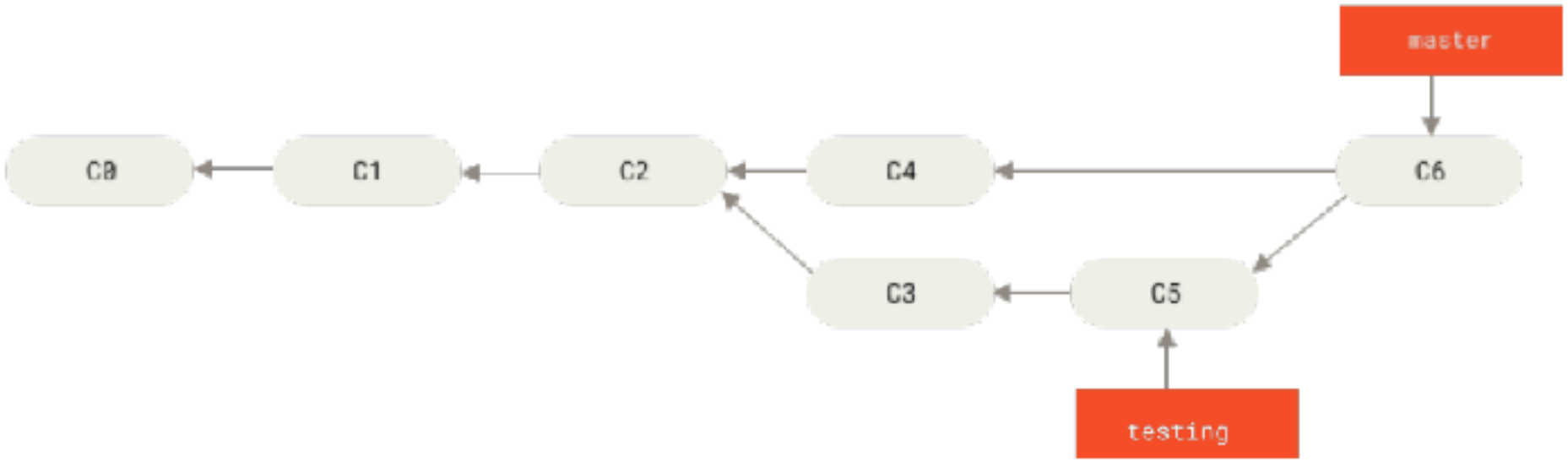
How git merge works

```
$ git merge testing
```



How git merge works

```
$ git merge testing
```

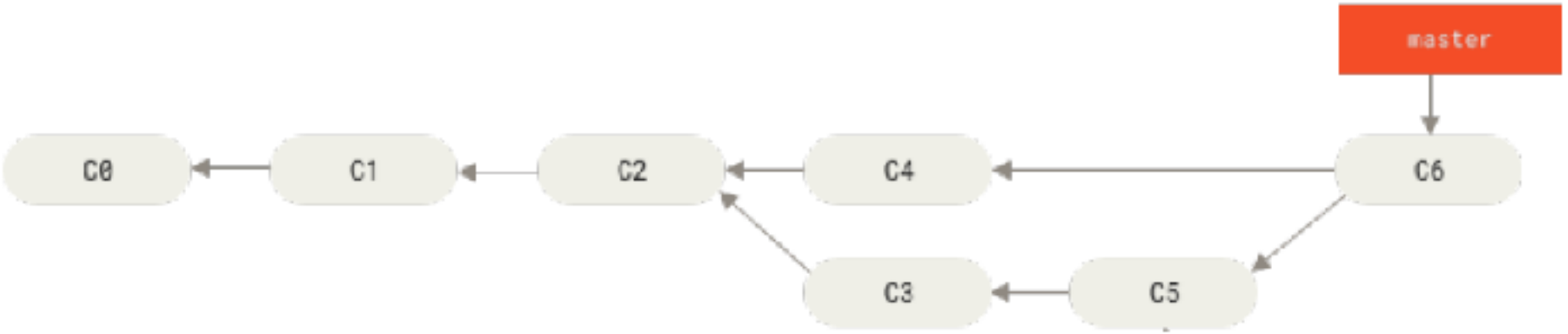


Common Workflow

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

How to delete branches

```
$ git branch -d testing
```

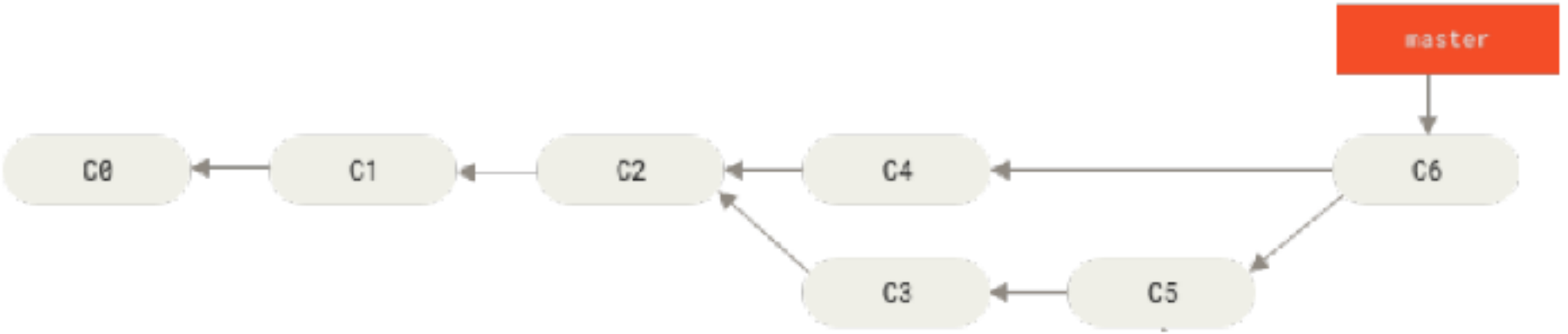


Common Workflow

1. Create temp local branch
2. Checkout temp branch
3. Edit/Add/Commit on temp branch
4. Checkout master branch
5. Pull to update master branch
6. Merge temp branch with updated master
7. Delete temp branch
8. Push to update server repos

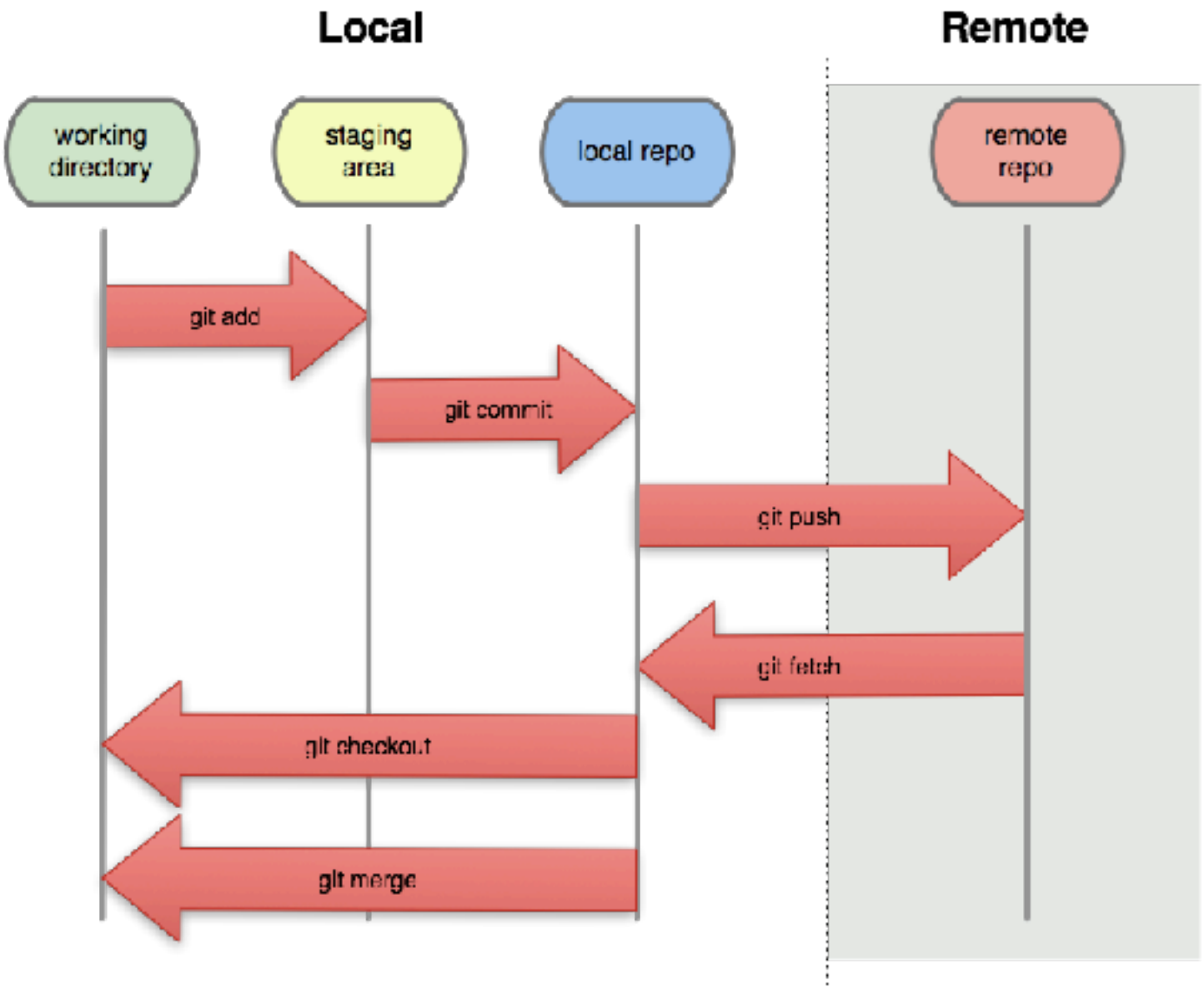
How git push works

```
$ git push
```



Should update server repos

(if no one else has pushed commits to master branch since last pull)


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

Tips

- git output contains lots of hints
  - git status is your friend!
- Merging may not be as easy as I showed
  - E.g.: Multiple collabs updated same parts of file
  - See Pro Git 3.2
- Pull before starting temp branch
- Team communication important!