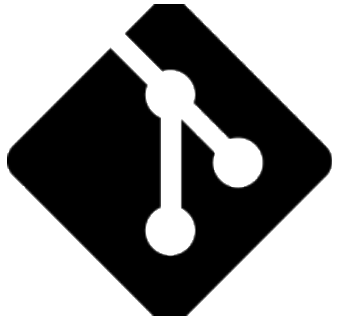
Programming workshop

Version Control Systems

Part I



git

5:15 pm

JetBrains student program

jetbrains.com/student

Intellij IDEA

jetbrains.com/idea

Pycharm

jetbrains.com/pycharm

PhpStorm

jetbrains.com/phpstorm

And the list goes on…

PRO Git v2

https://git-scm.com/book/en/v2

Graphical content in this demonstration in partial is taken from PRO Git v2 book.

Version control

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Version control

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Dummy version control

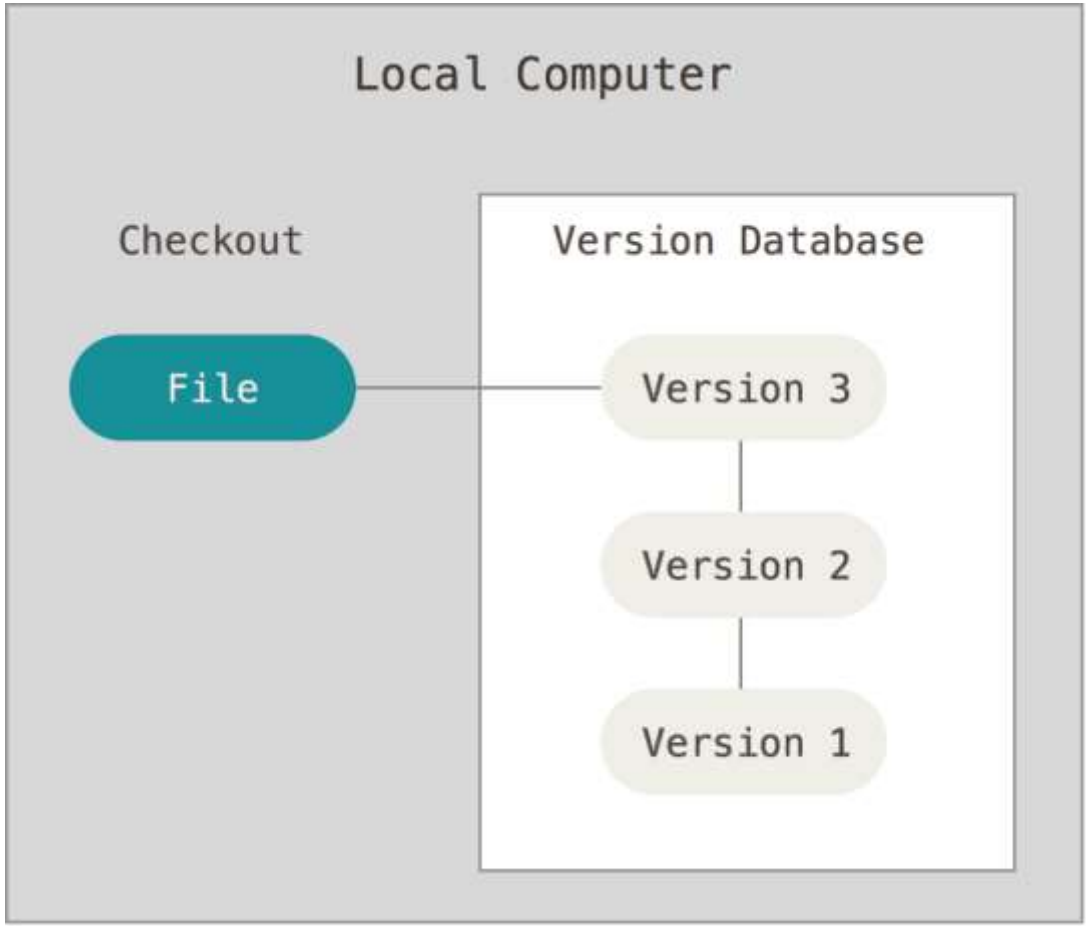
copy files into another directory
(perhaps a time-stamped directory, if they're clever).

Dummy version control

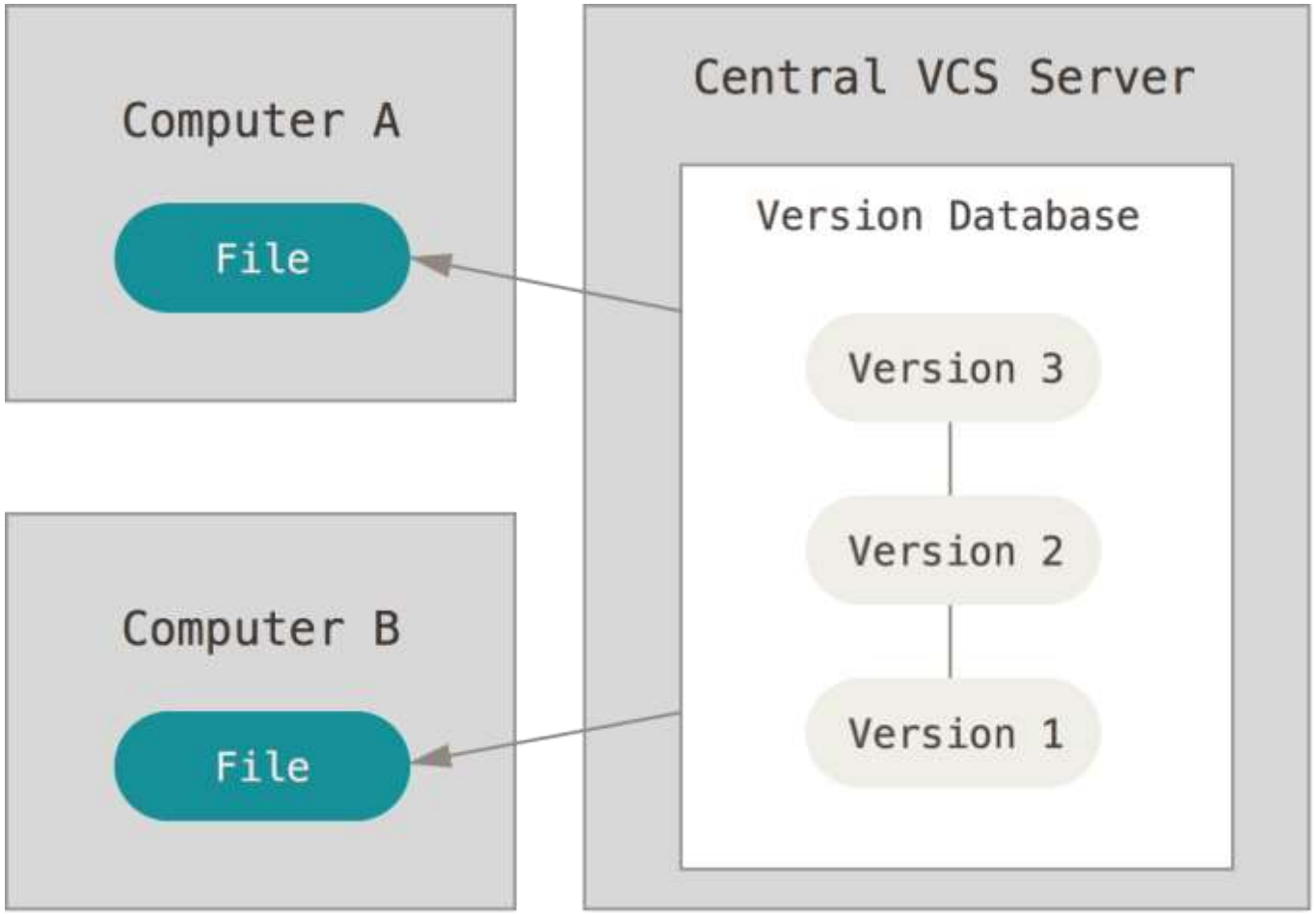
copy files into another directory
(perhaps a time-stamped directory, if they're clever).

It is easy to forget which directory you're in and accidentally write to the wrong file or copy over files you don't mean to.

Local version control



Centralized version control (CVCS)



Centralized version control (CVCS)

Usual suspects:

- CVS
- Subversion
- Perforce

Advantages:

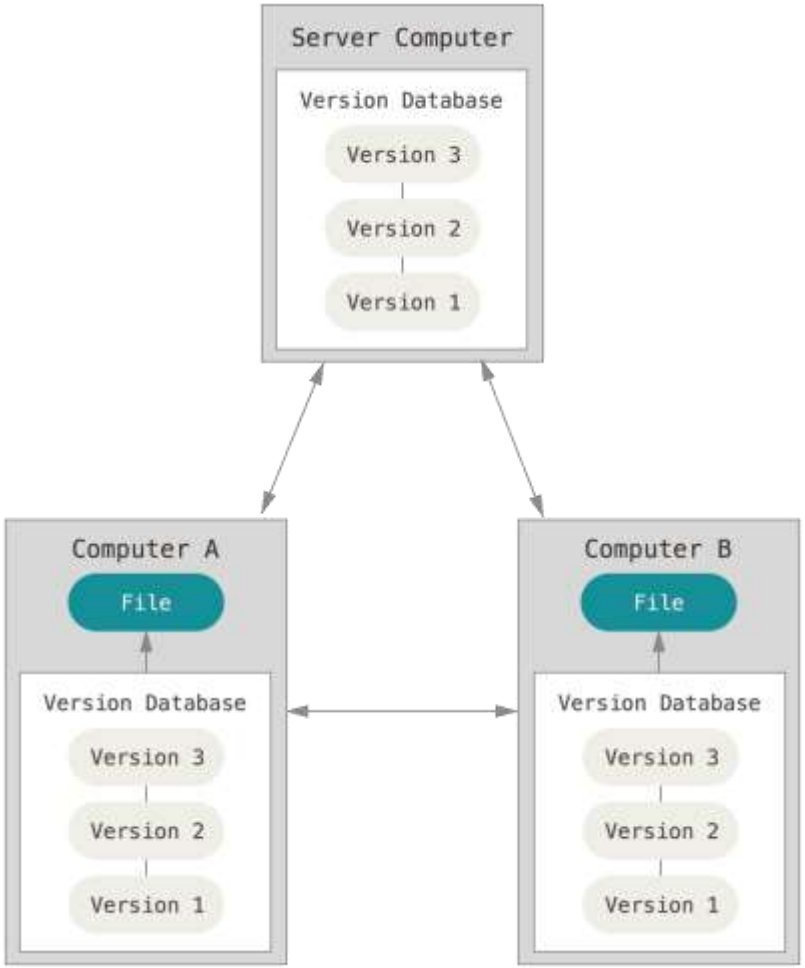
- everyone knows to a certain degree what everyone else on the project is doing
- administrators have fine-grained control over who can do what
- it is far easier to administer a CVCS than it is to deal with local databases on every client

Centralized version control (CVCS)

Disadvantages:

- **Single point of failure** (If the main server goes down the developers can't save versioned change)
- Remote commits are slow
- Unsolicited changes might ruin the development
- **Single point of failure** (if the hard disk of the central databse becomes corrupted the entire history could be lost)
- No easy way to go from single-developer to multipl-developer state

Distributed version control (DVCS)



Distributed version control (DVCS)

Usual suspects:

- Git
- Mercurial
- Bazaar

Advantages:

- Full history is available to everyone at all times
- Extemely fast due to local nature of the majority of the operation
- No access to remote server is required
- Sharing can be done among any subset of developers, before making changes public
- Branching and merging is SUPER easy

Distributed version control (DVCS)

Disadvantages:

- Large number of files that can not be easily compressed may be an issue
- Very long project history may result in a rather long initial download

Git

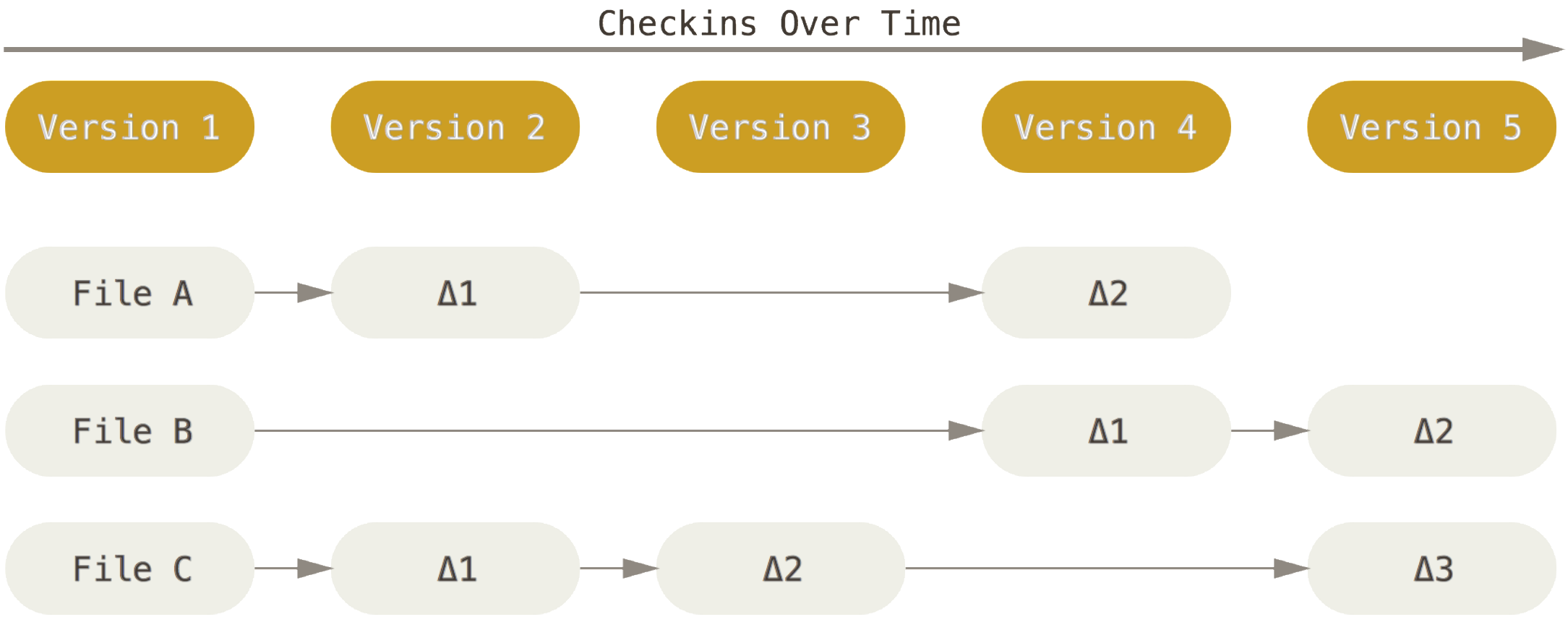
Date of birth: 2005

Creator: Linus Torvalds

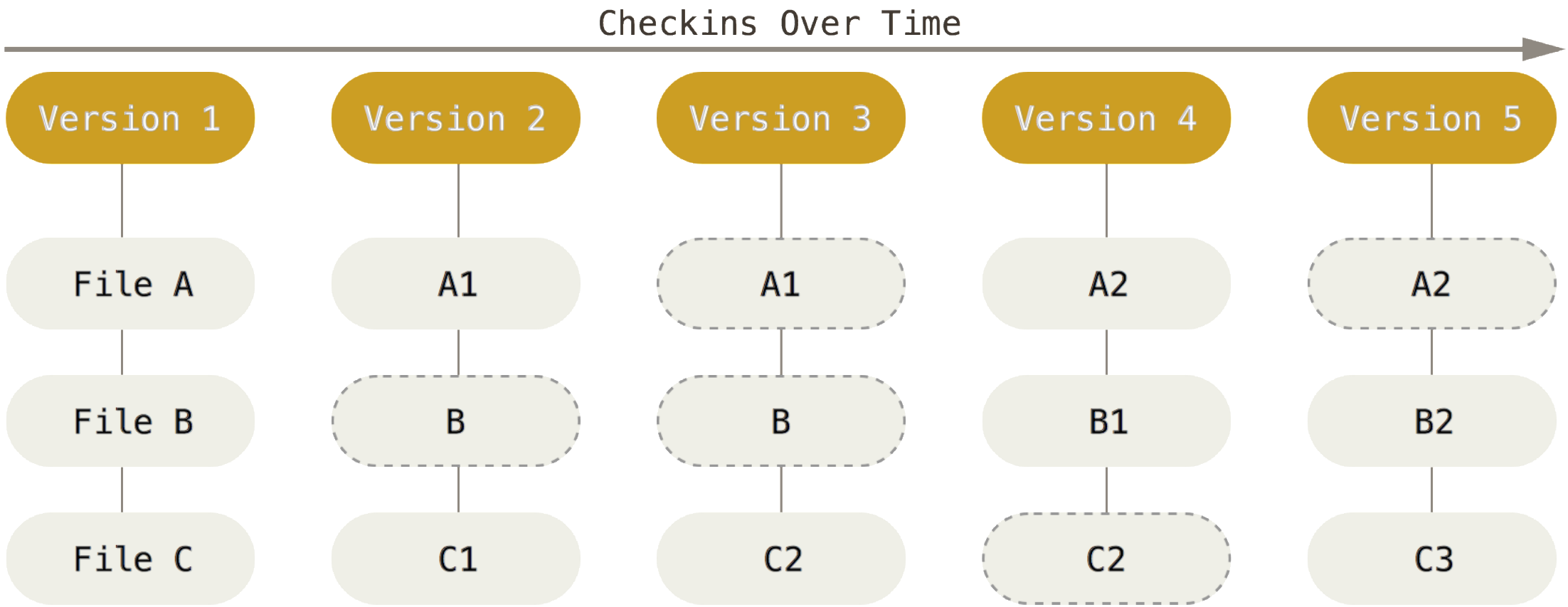
Requirements:

- Speed
- Simple design
- Strong support fo non-linear development (thoudands of paraller branches)
- Fully distributed
- Large projects support

Git: snapshots, not differences



Git: snapshots, not differences



Git

- Filesystem on steroids

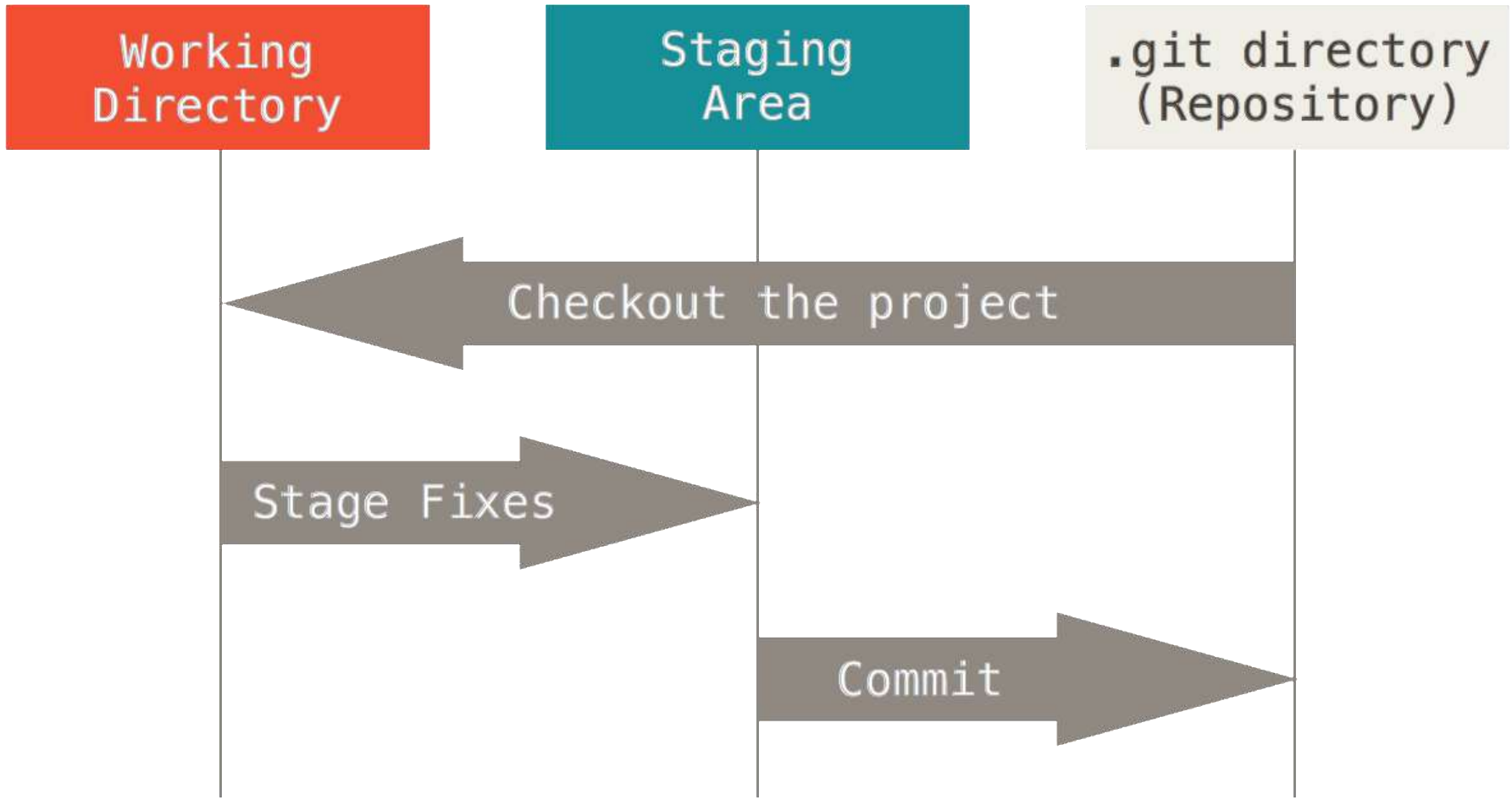
Git

- Filesystem on steroids
- Almost every operation is local

Git

- Filesystem on steroids
- Almost every operation is local
- Integrity
- Only adding data throughout time

Git: project structure



Git: basic workflow

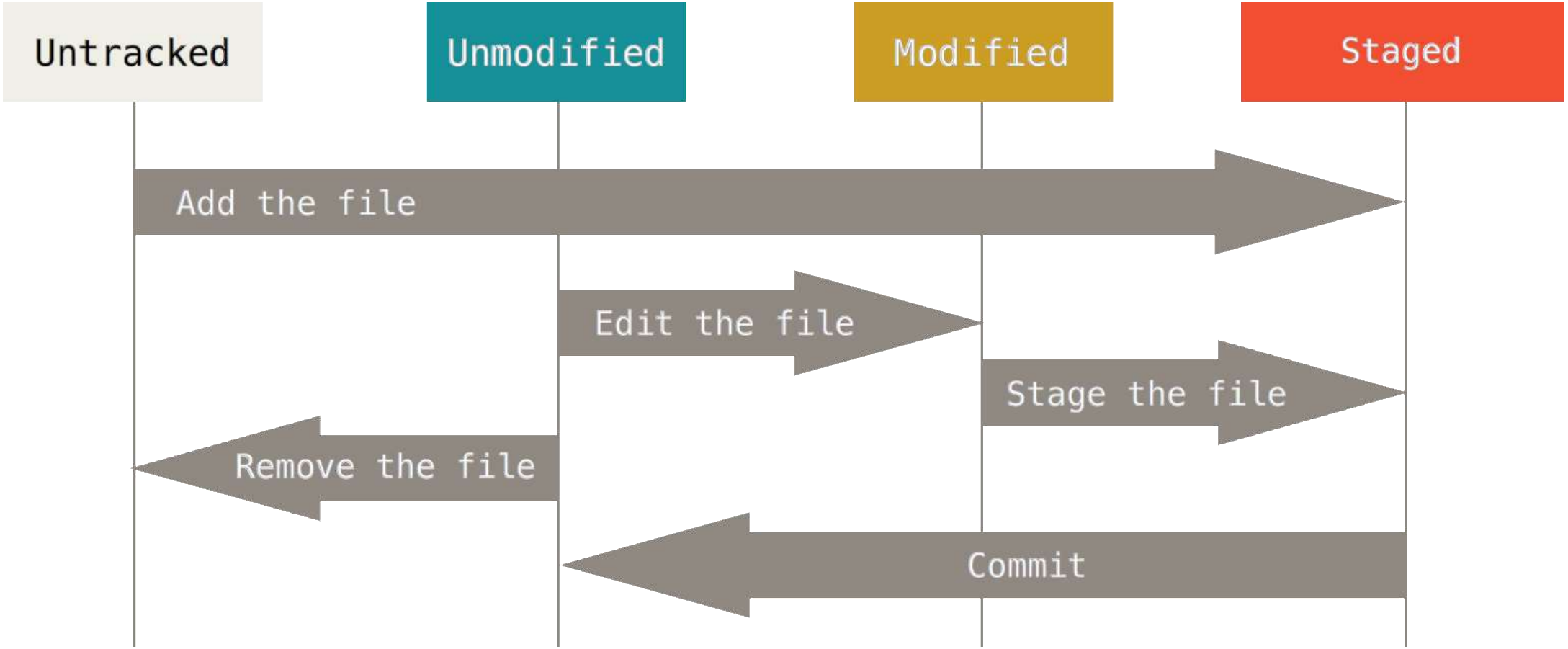
1. Checkout a revision to work on from **repository**

2. Modify files in **working directory**

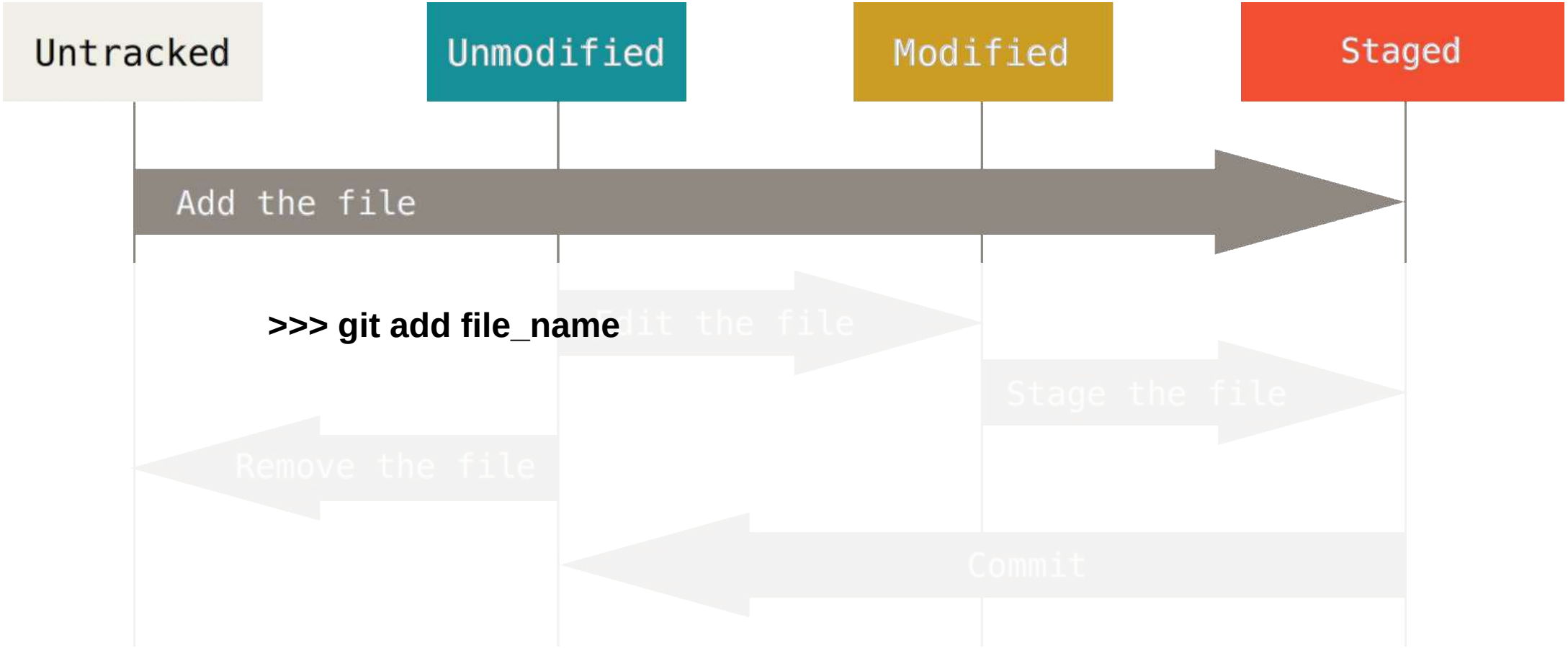
3.* Add modified files into the **staging area** as you go

4. **Commit** you staged files into the repository

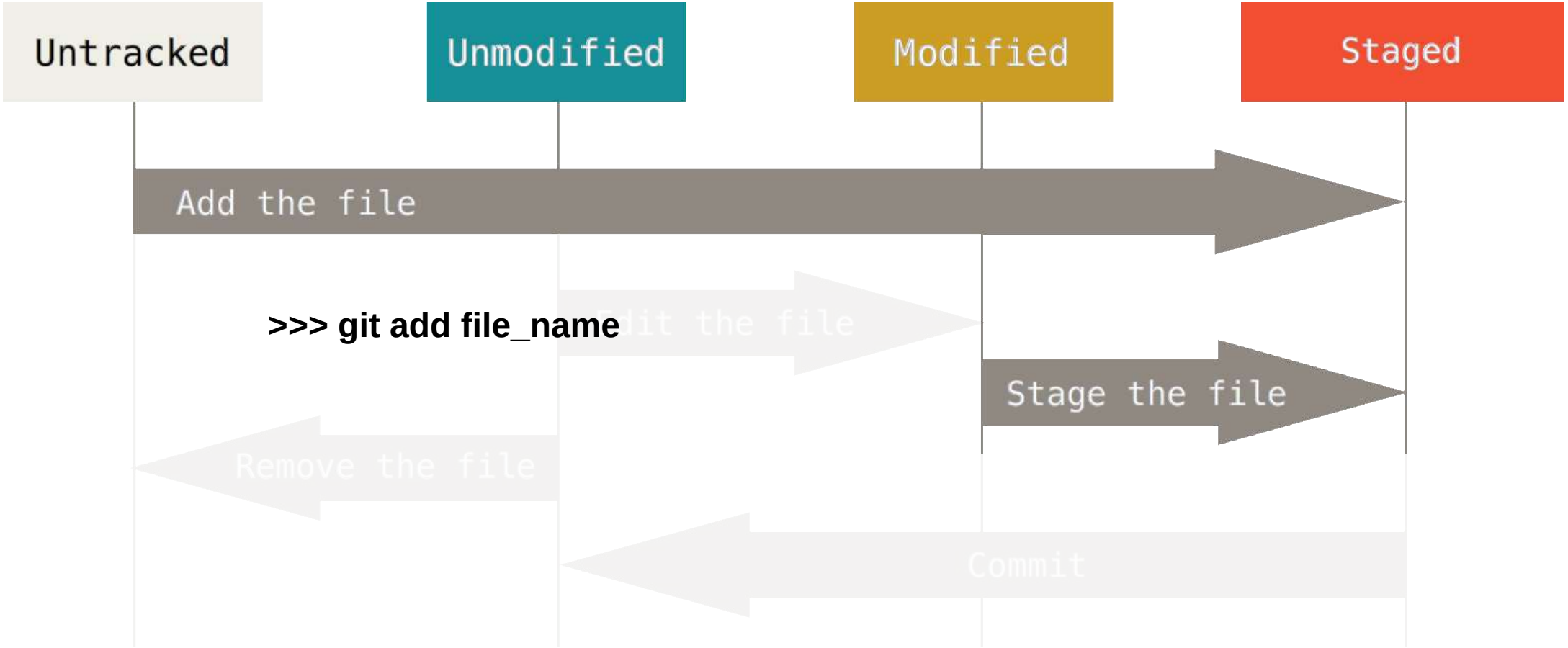
Git: files status lifecycle



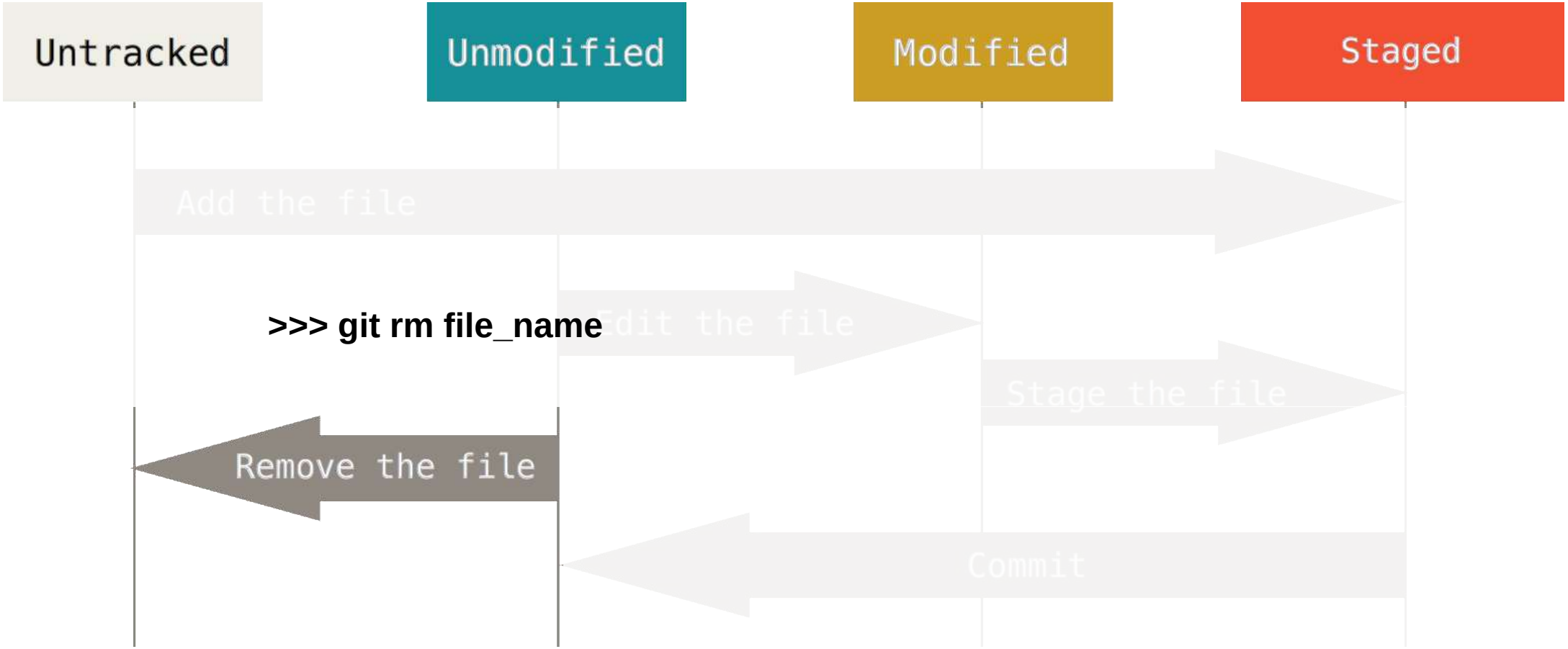
Git: files status lifecycle



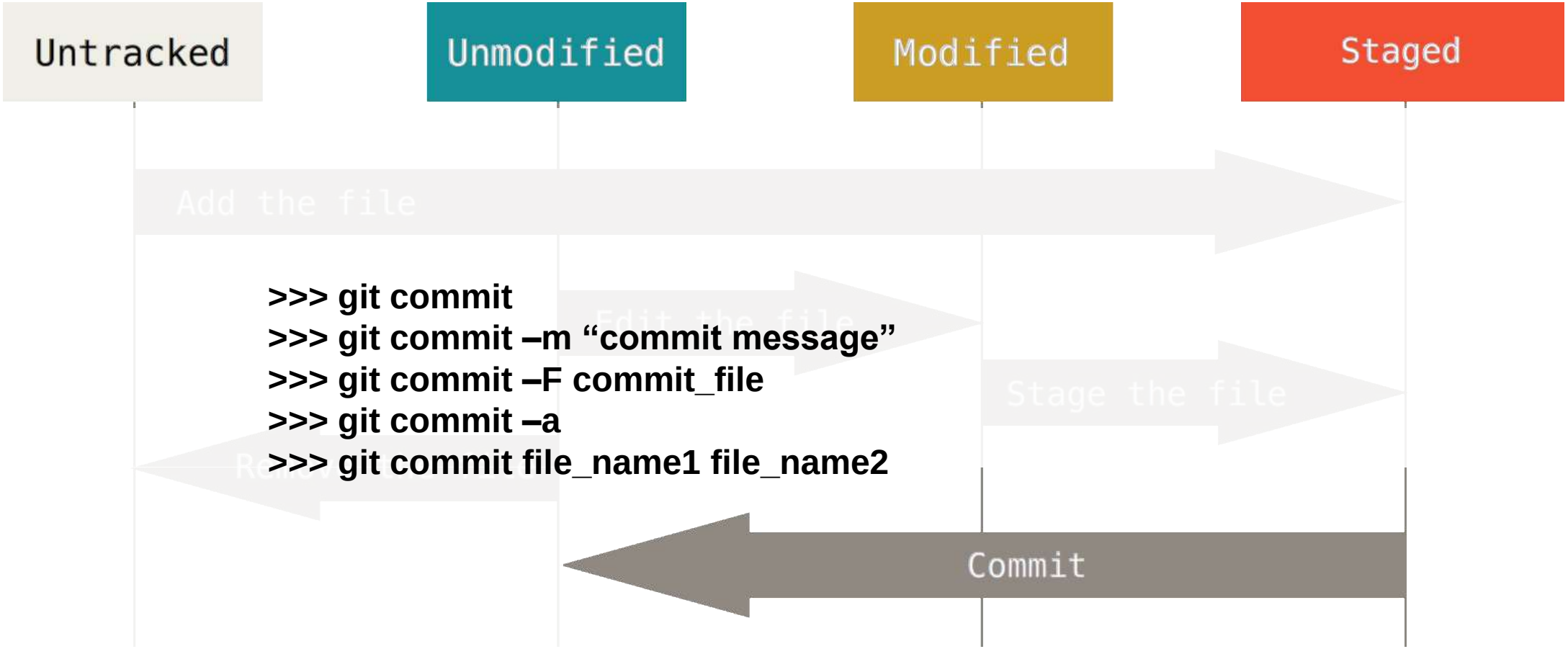
Git: files status lifecycle



Git: files status lifecycle

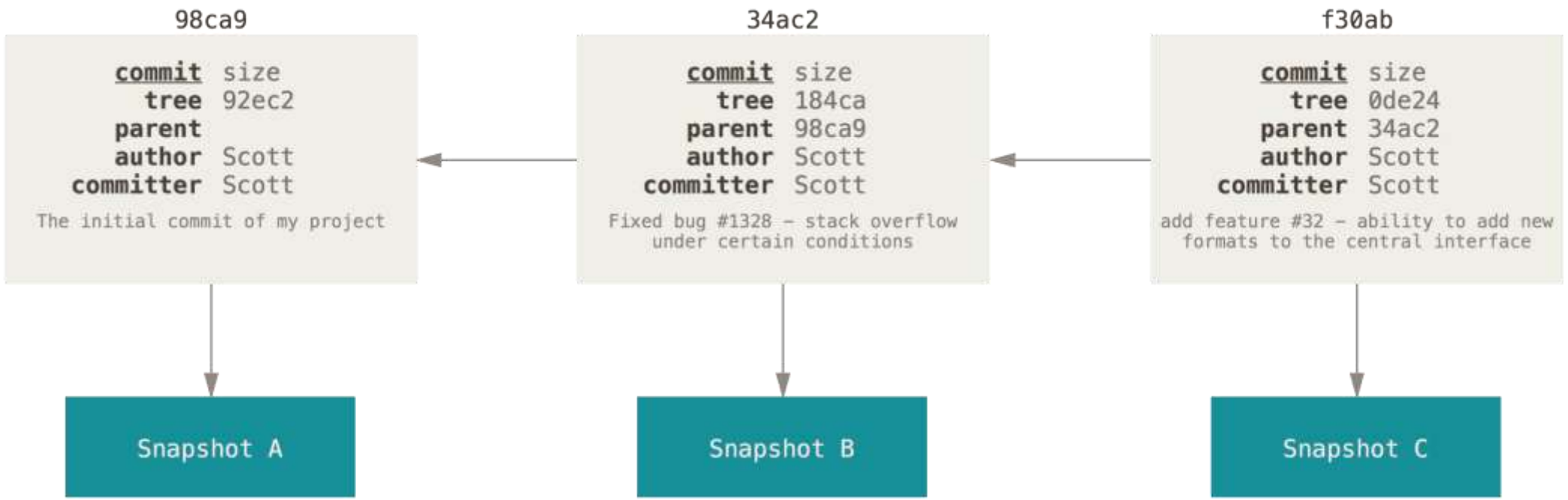


Git: files status lifecycle

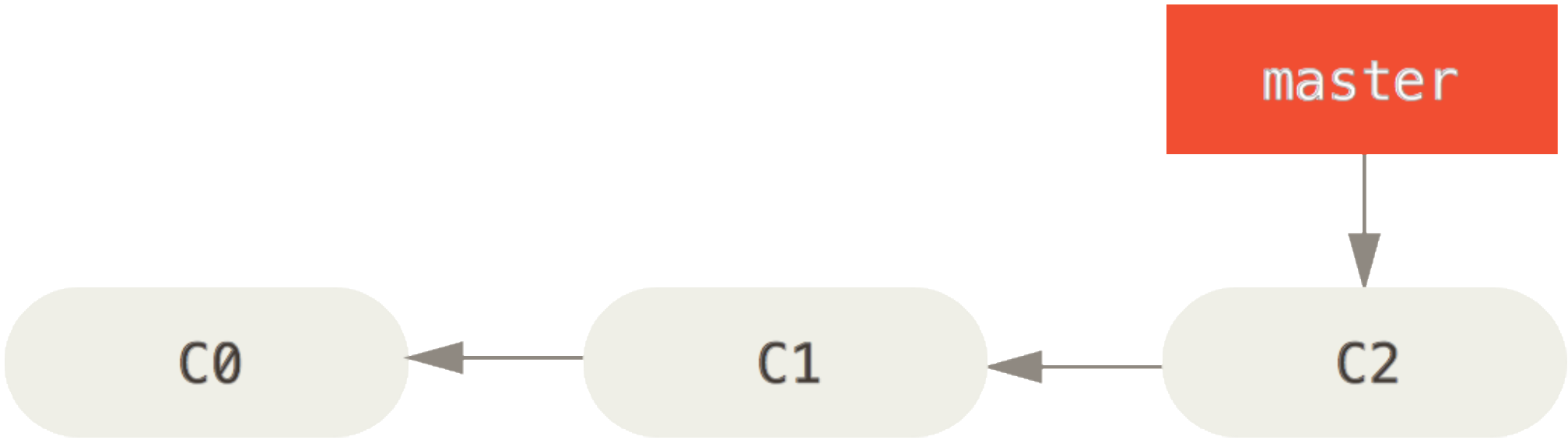


>>> git commit
>>> git commit –m “commit message”
>>> git commit –F commit_file
>>> git commit –a
>>> git commit file_name1 file_name2

Git: branch


98ca9
commit size
tree 92ec2
parent
author Scott
committer Scott
The initial commit of my project
34ac2
commit size
tree 184ca
parent 98ca9
author Scott
committer Scott
Fixed bug #1328 - stack overflow under certain conditions
f30ab
commit size
tree 0de24
parent 34ac2
author Scott
committer Scott
add feature #32 - ability to add new formats to the central interface
Snapshot A
Snapshot B
Snapshot C

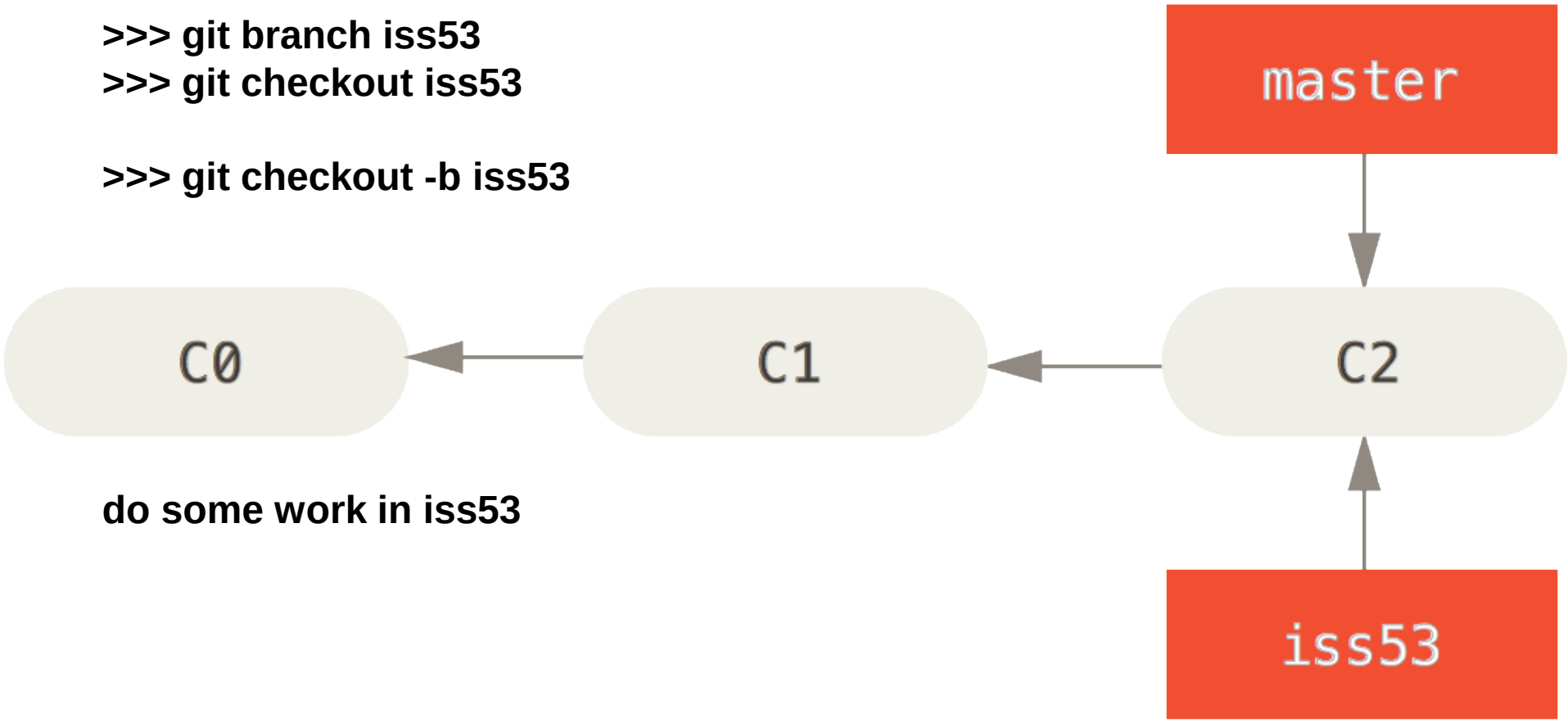
Git: branches



Git: branches

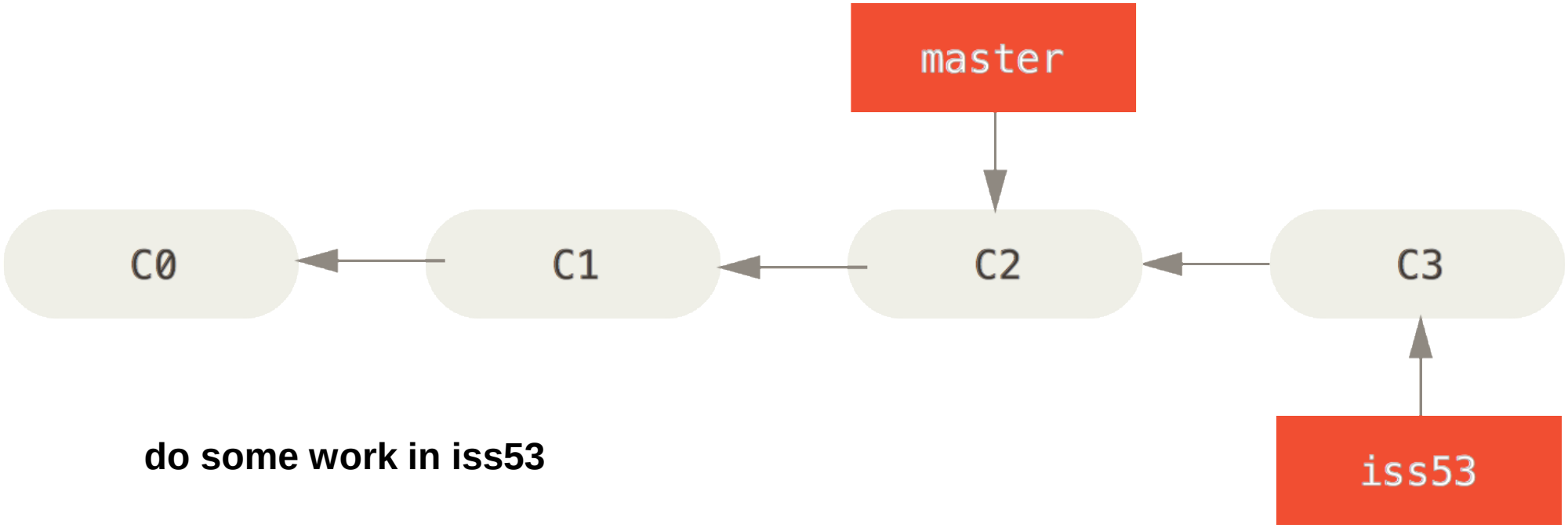
>>> git branch iss53
>>> git checkout iss53

>>> git checkout -b iss53



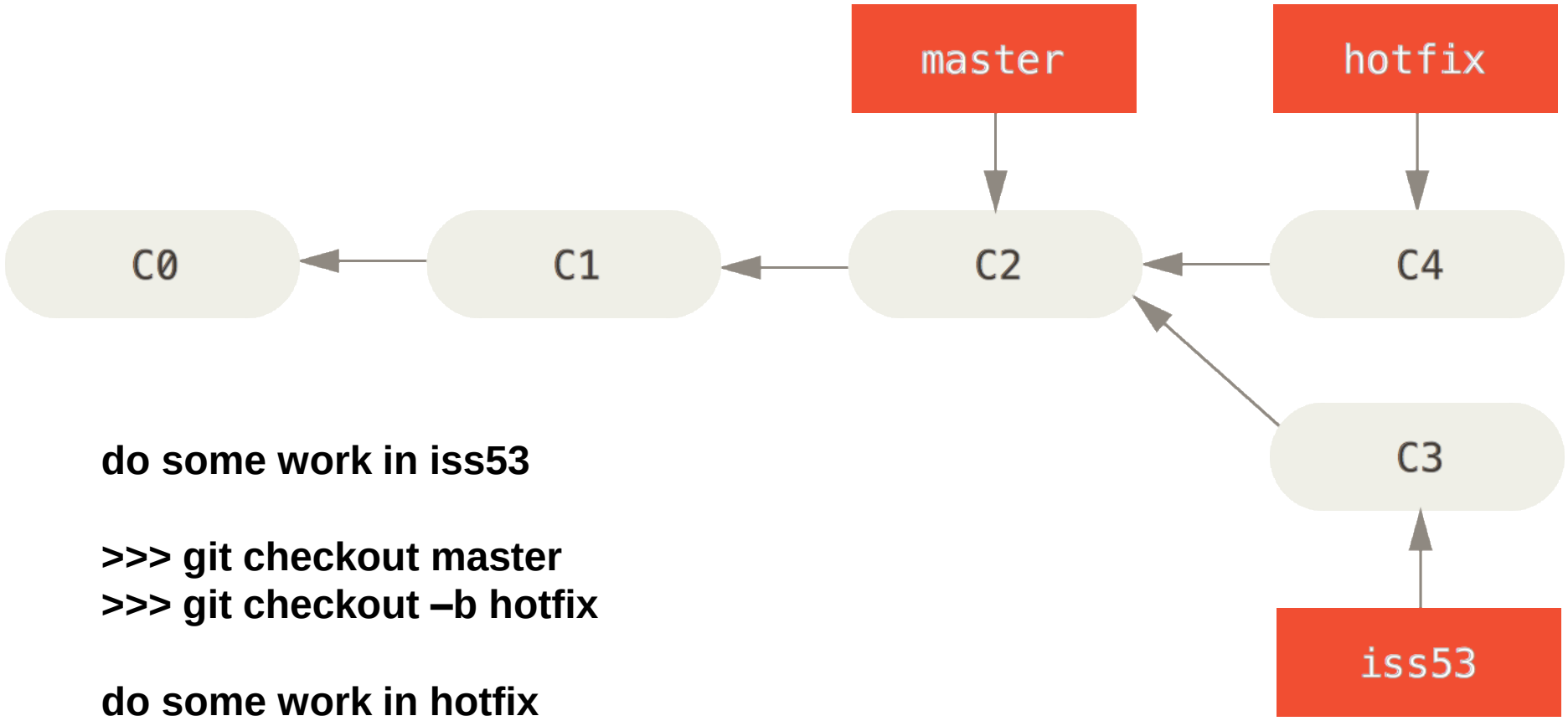
do some work in iss53

Git: branches



do some work in iss53

Git: branches

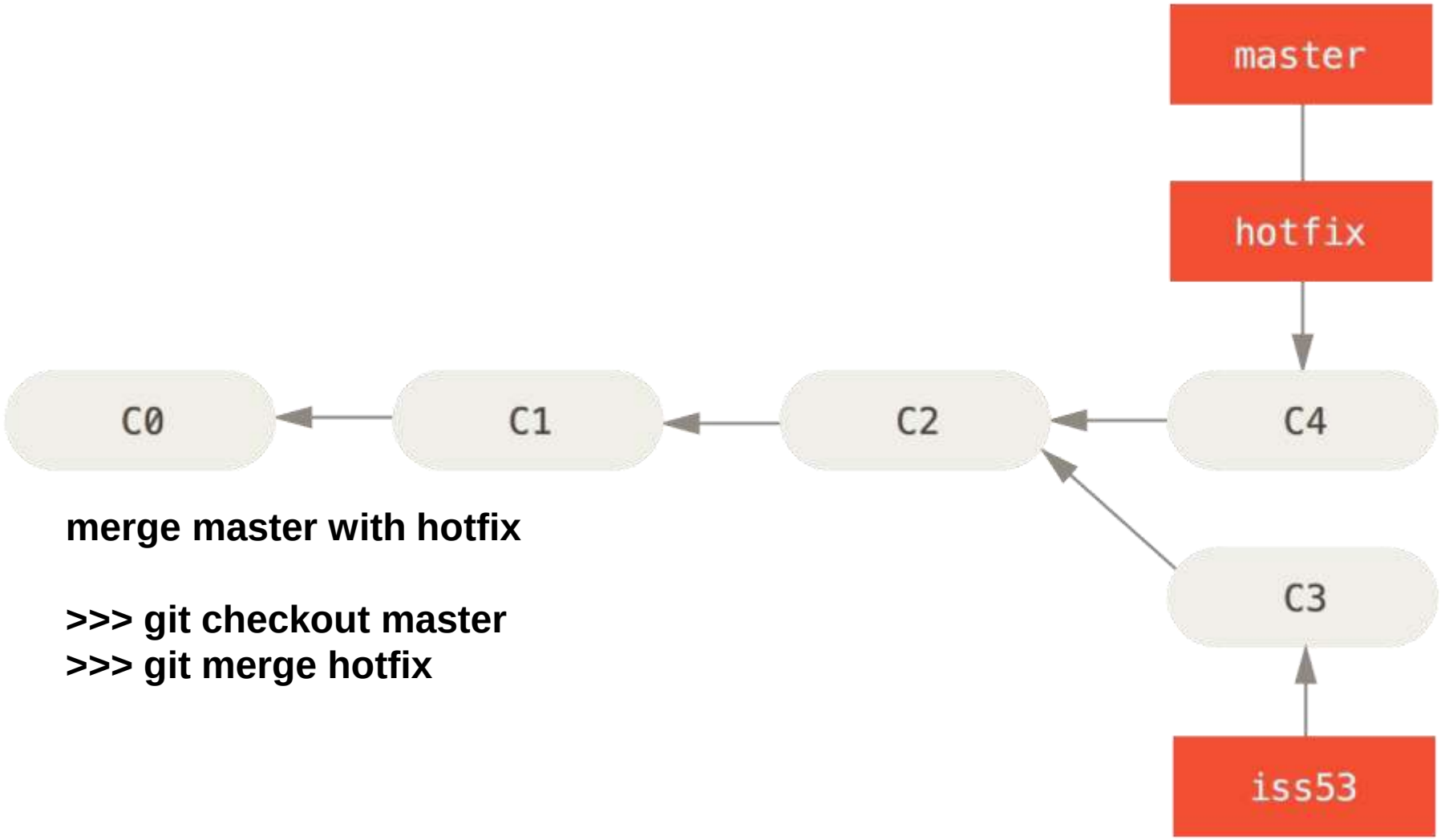


do some work in iss53

>>> git checkout master
>>> git checkout –b hotfix

do some work in hotfix

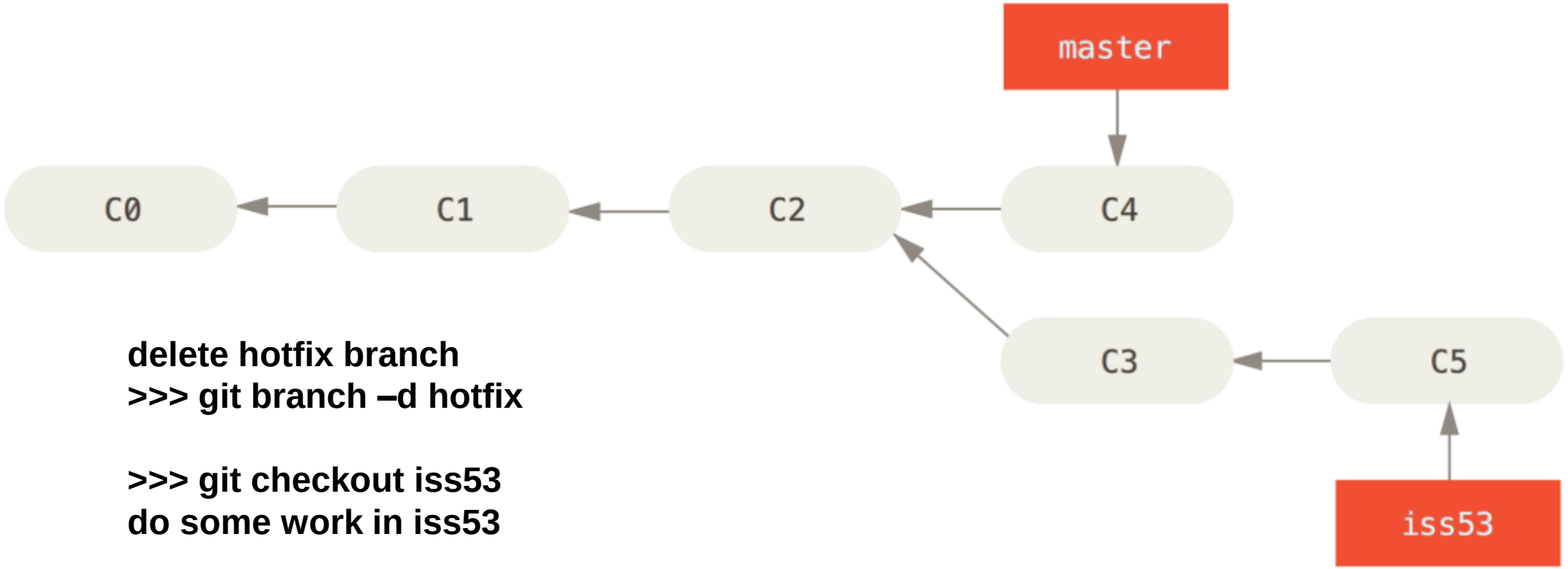
Git: branches



merge master with hotfix

>>> git checkout master
>>> git merge hotfix

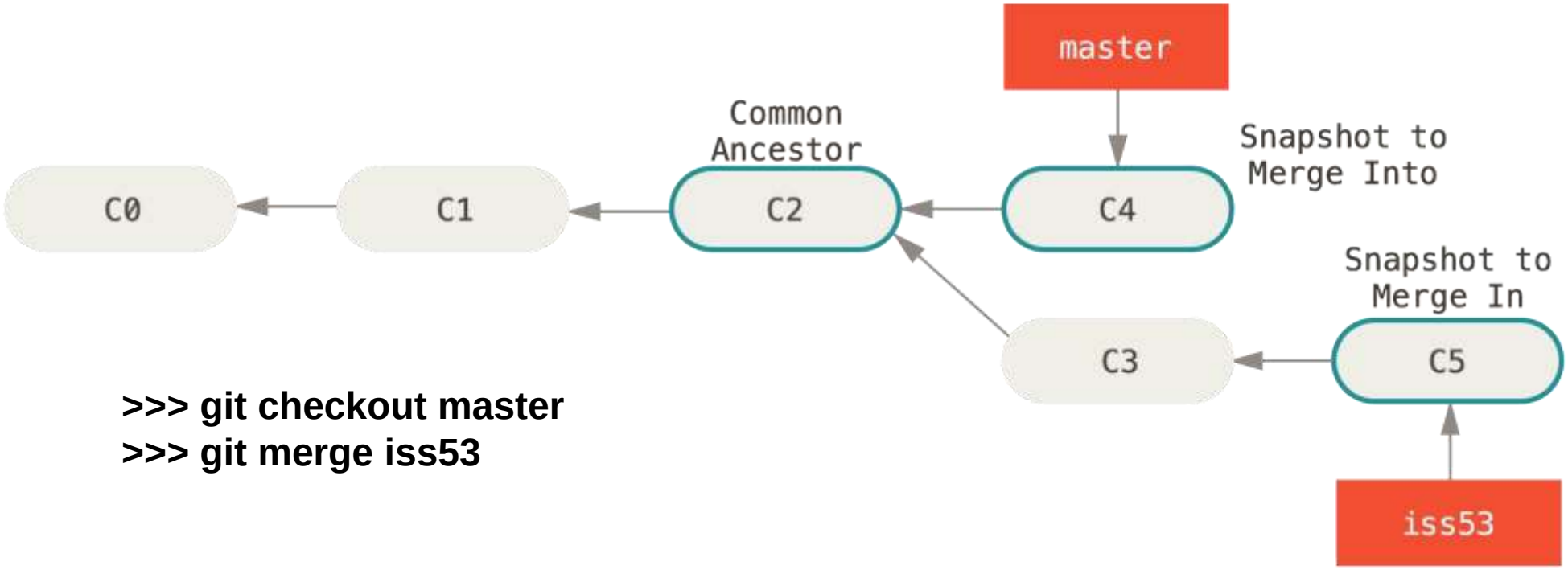
Git: branches



delete hotfix branch
>>> git branch –d hotfix

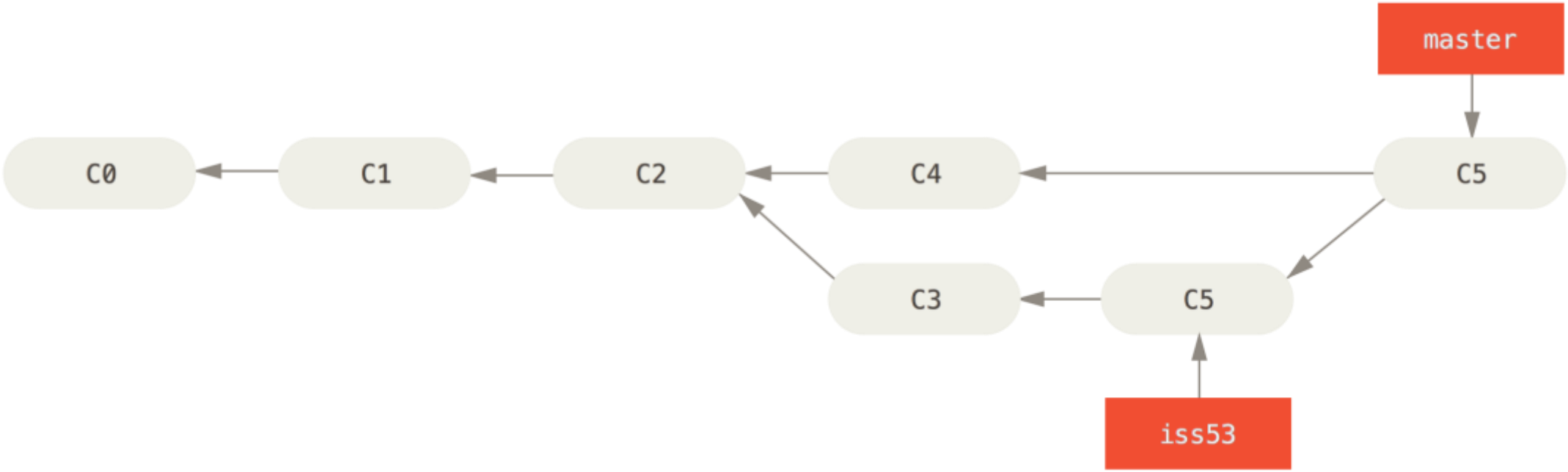
>>> git checkout iss53
do some work in iss53

Git: branches

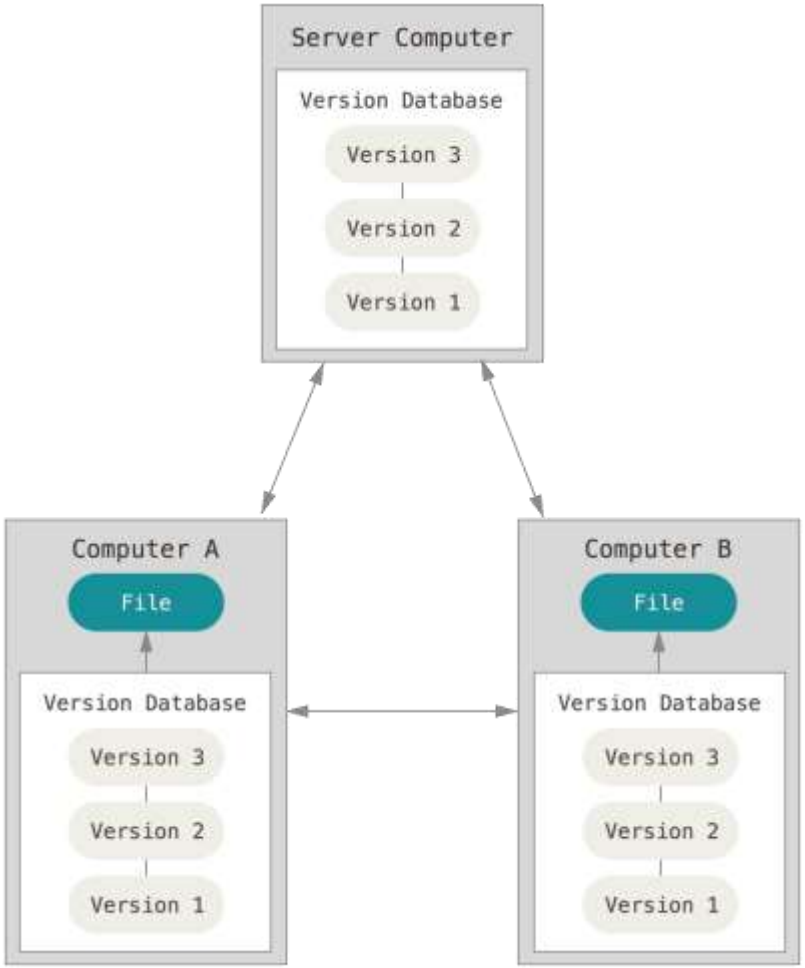


>>> git checkout master
>>> git merge iss53

Git: branches



Git: collaborating



Git: remotes

Adding a remote:

>>> git remote add github https://github.com/aganezov/vcs-sample.git

>>> git remote
github

>>> git remote –v
github https://github.com/aganezov/vcs-sample.git (fetch)
github https://github.com/aganezov/vcs-sample.git (push)

Git: remotes

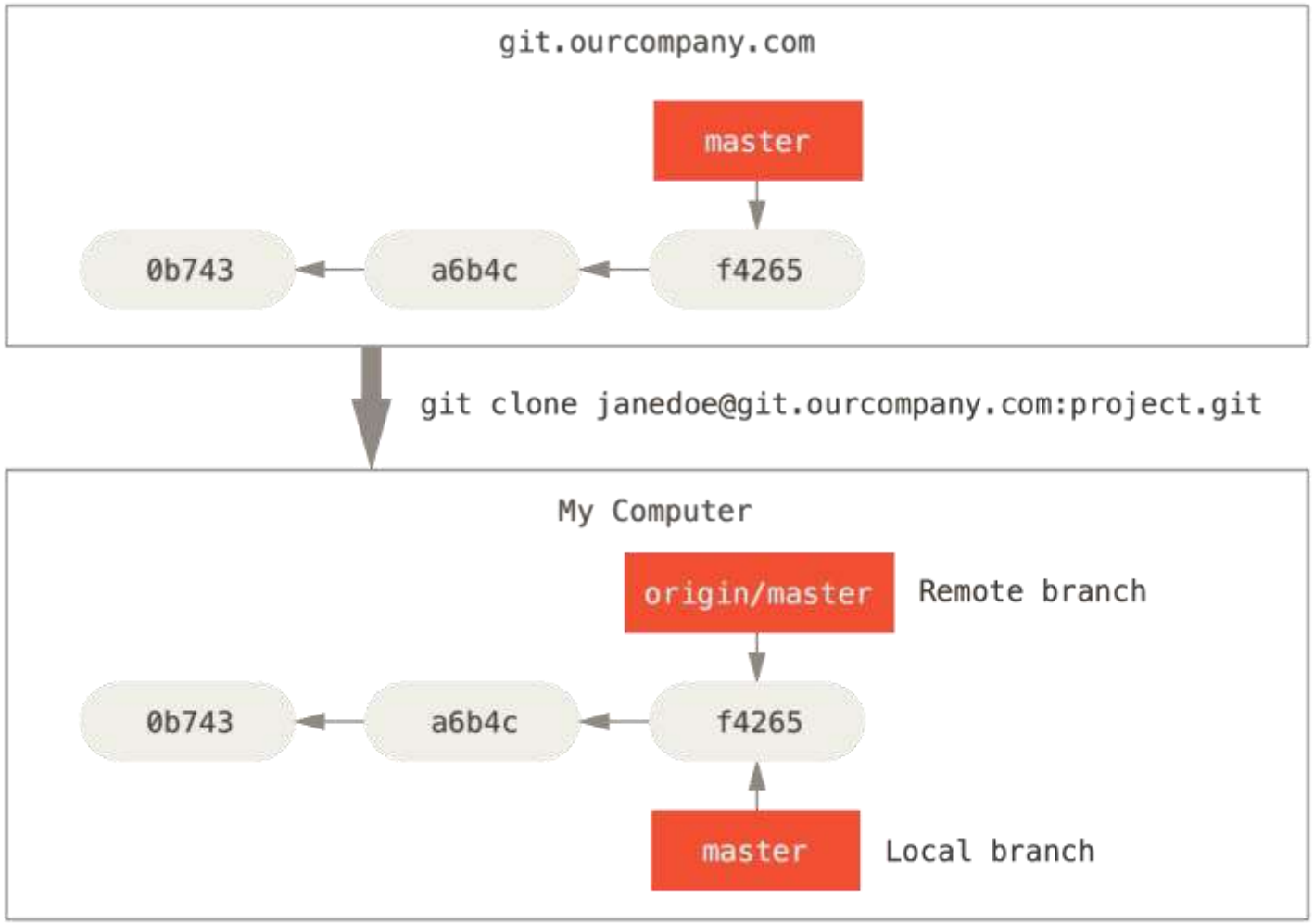
Remote branches are references of states of branches on your remote repository.

You can think of them as local branches that you can not move. They are mvoed automatically due to network communication.

They take form of **(remote)/(branch)**

>>> git remote

Git: remotes


git.ourcompany.com
master
0b743
a6b4c
f4265
git clone janedoe@git.ourcompany.com:project.git
My Computer
origin/master
Remote branch
0b743
a6b4c
f4265
master
Local branch

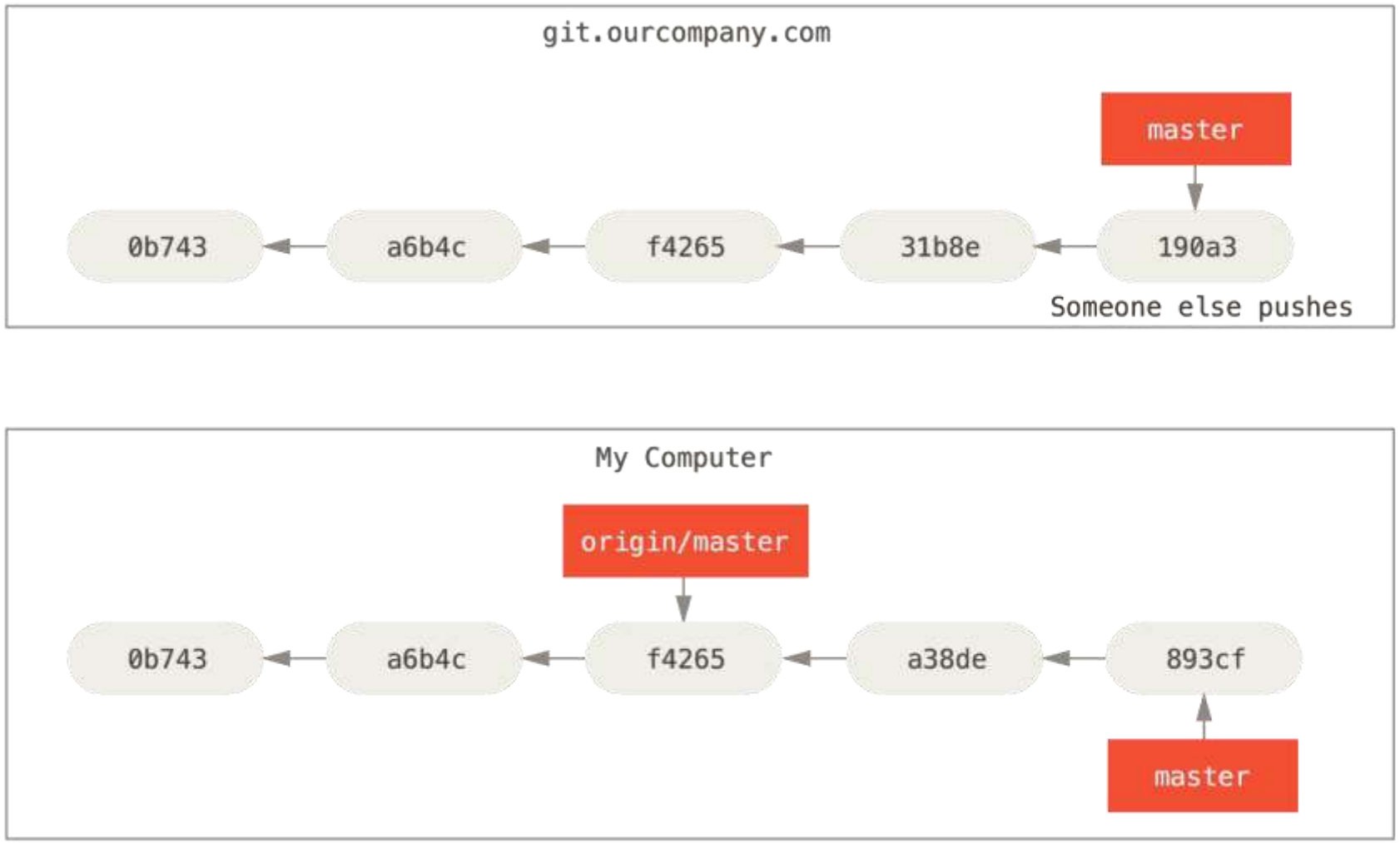
Git: remotes

- **>>> git push github master [--set-upstream]**
- **>>> git push**

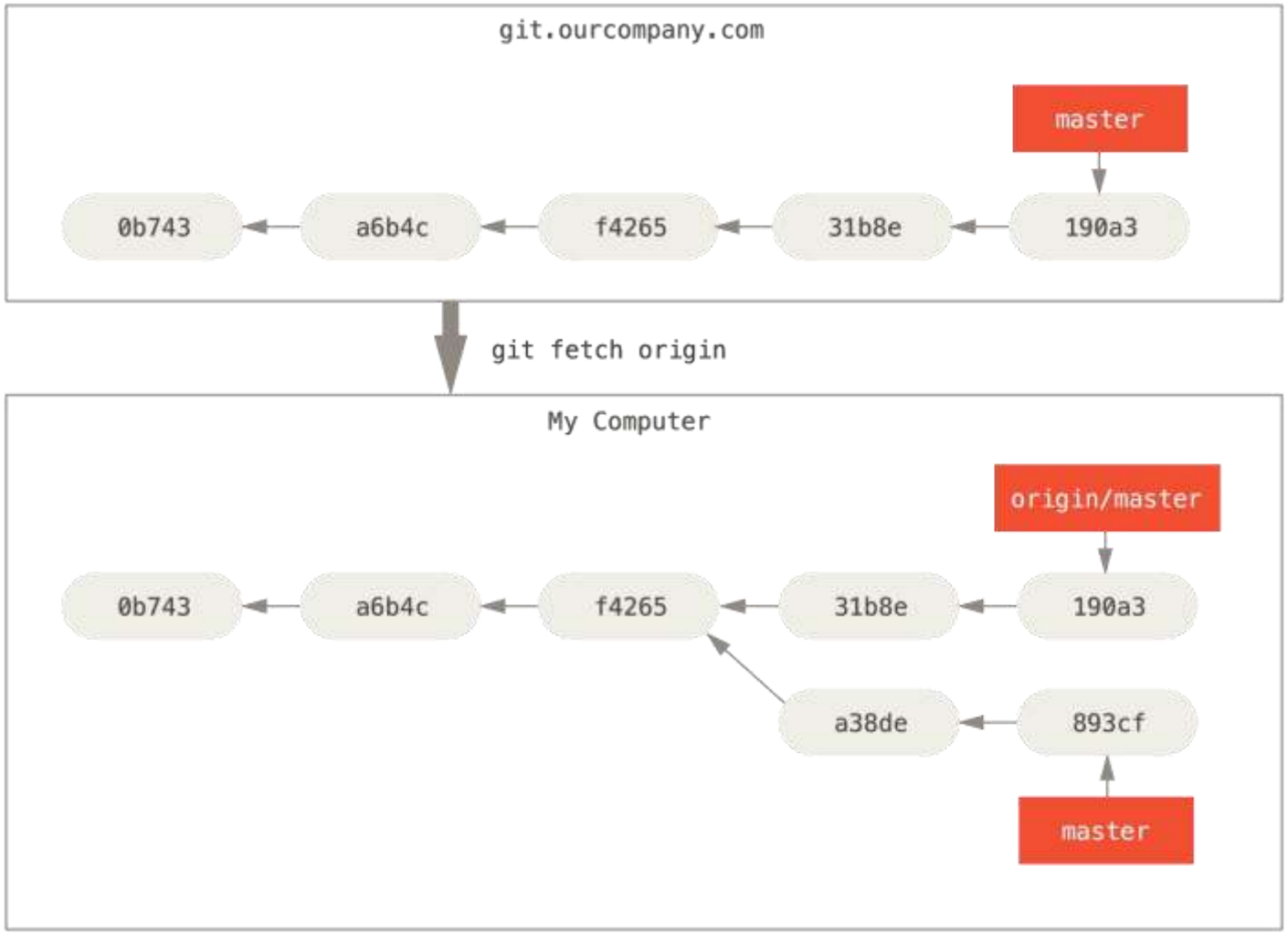
- **>>> git fetch**
  updates the (remote)/(branch) pointers for you repository

- **>>> git pull**
  >>> git fetch
  >>> git merge
  merges remote tracked branch into current one

Git: remotes


git.ourcompany.com
master
0b743
a6b4c
f4265
31b8e
190a3
Someone else pushes
My Computer
origin/master
0b743
a6b4c
f4265
a38de
893cf
master

Git: remotes


git.ourcompany.com
master
0b743
a6b4c
f4265
31b8e
190a3
git fetch origin
My Computer
origin/master
0b743
a6b4c
f4265
31b8e
190a3
a38de
893cf
master

Takeaways:

- VCS can save you from really bad things
- Version Control Systems are easy to use

- Git is easy to use
- Every Git revision is the snapshot of the **entire** project state!
- Git branch is a simple pointer to a revision
- Git remote branch is another simple pointer

- You are where the HEAD is
- Branching and merging is super easy

- Github / Bitbucket are good free Git remote server options

Next **JetBrains GWU** meeting

“VCS – Git – Part II”

December 17, 5:00 – 7:00 pm SEH room 2000

Facebook
https://www.facebook.com/groups/jetbrains.gwu/
Twitter
https://twitter.com/JetBrains_GWU
Email
jetbrains.gwu@gmail.com
aganezov@gwu.edu