

CSE 403: Software Engineering, Spring 2015

courses.cs.washington.edu/courses/cse403/15sp/


Version Control

Emina Torlak

emina@cs.washington.edu

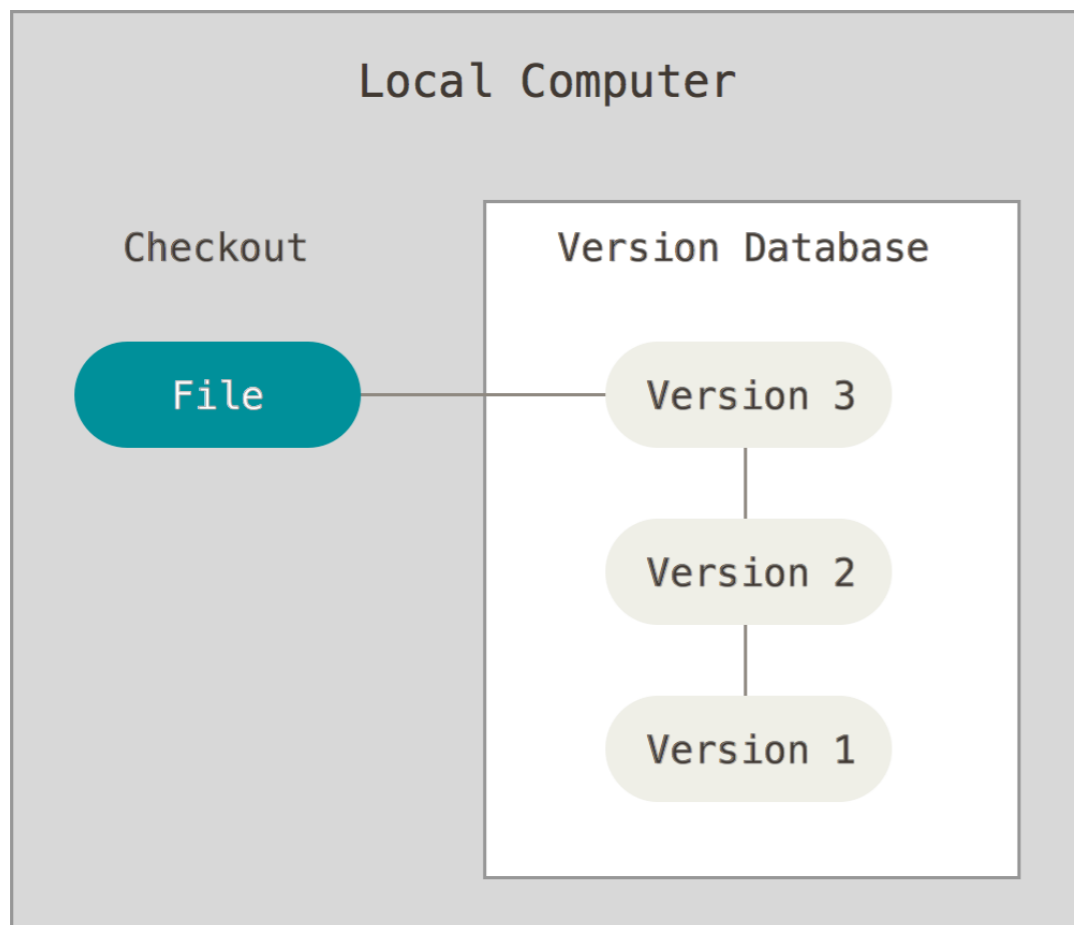
Outline

- Version control systems
- Basic concepts behind Git
- Working with Git
 - Creating or cloning a repository
 - Staging and committing changes
 - Pushing and pulling
 - Branching and merging
- Hints for effective use of Git



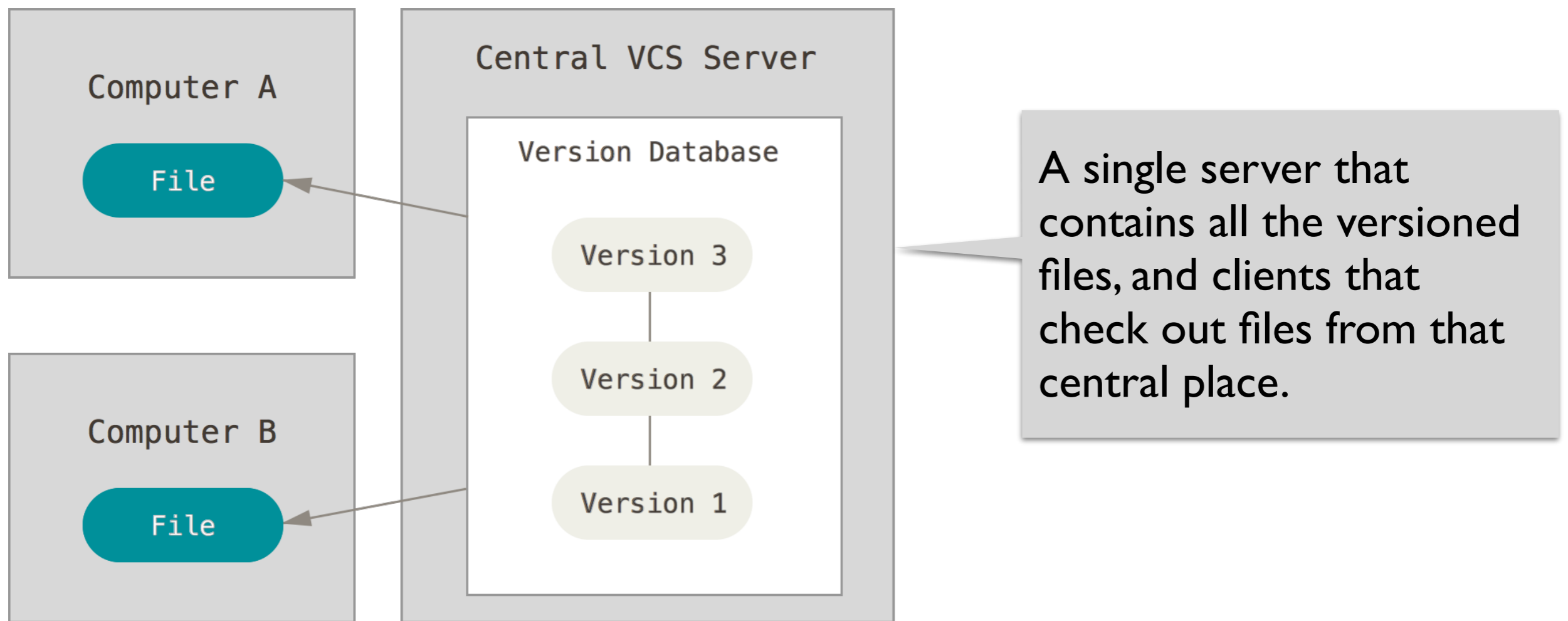
These slides are based on
<http://git-scm.com/book/en/>

Version control systems

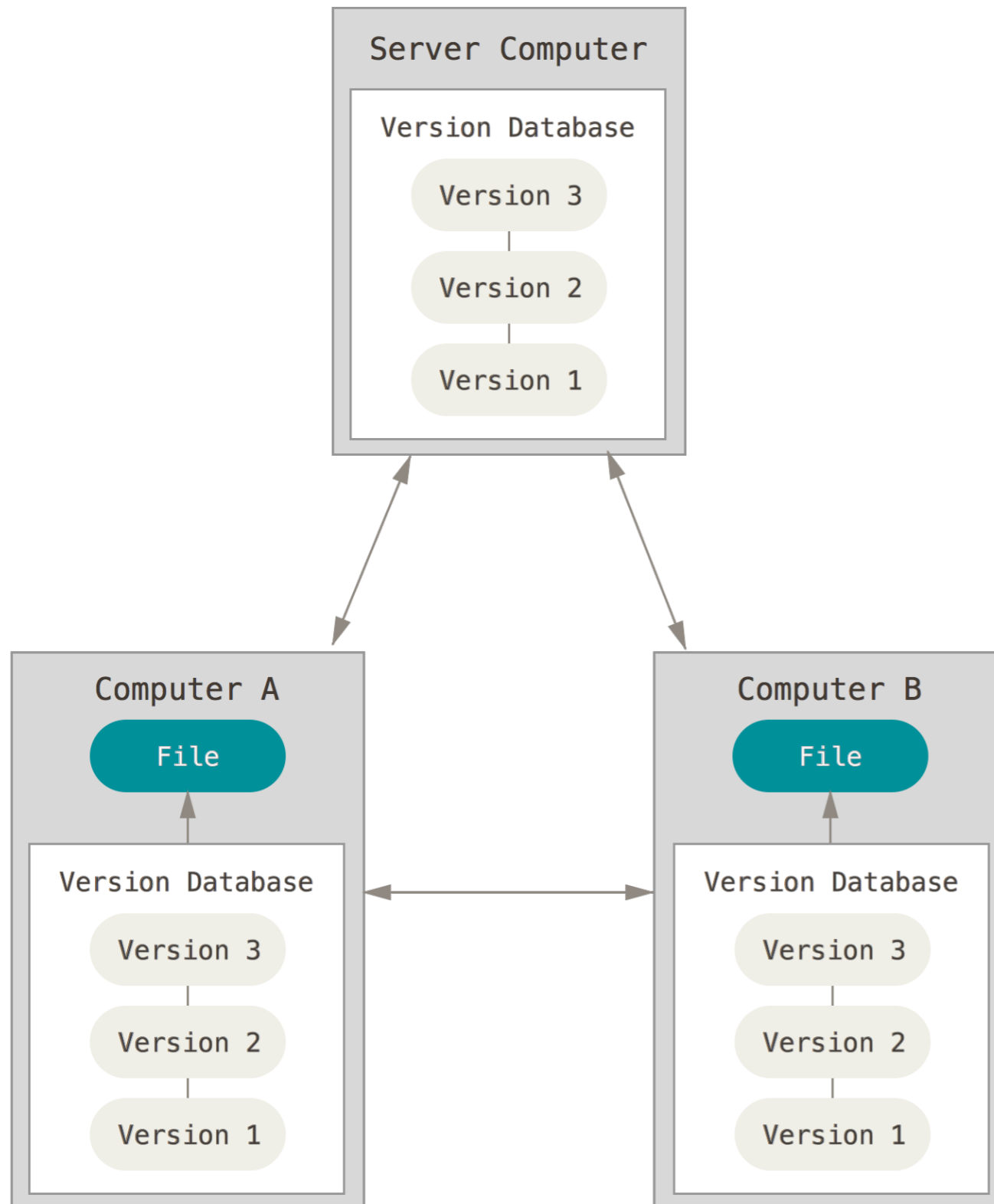


A system that records changes to a file or set of files over time so that you can recall specific versions later.

Centralized version control systems



Distributed version control systems



Clients fully mirror the repository (instead of just checking out the latest snapshot of the files).

A brief history of Git

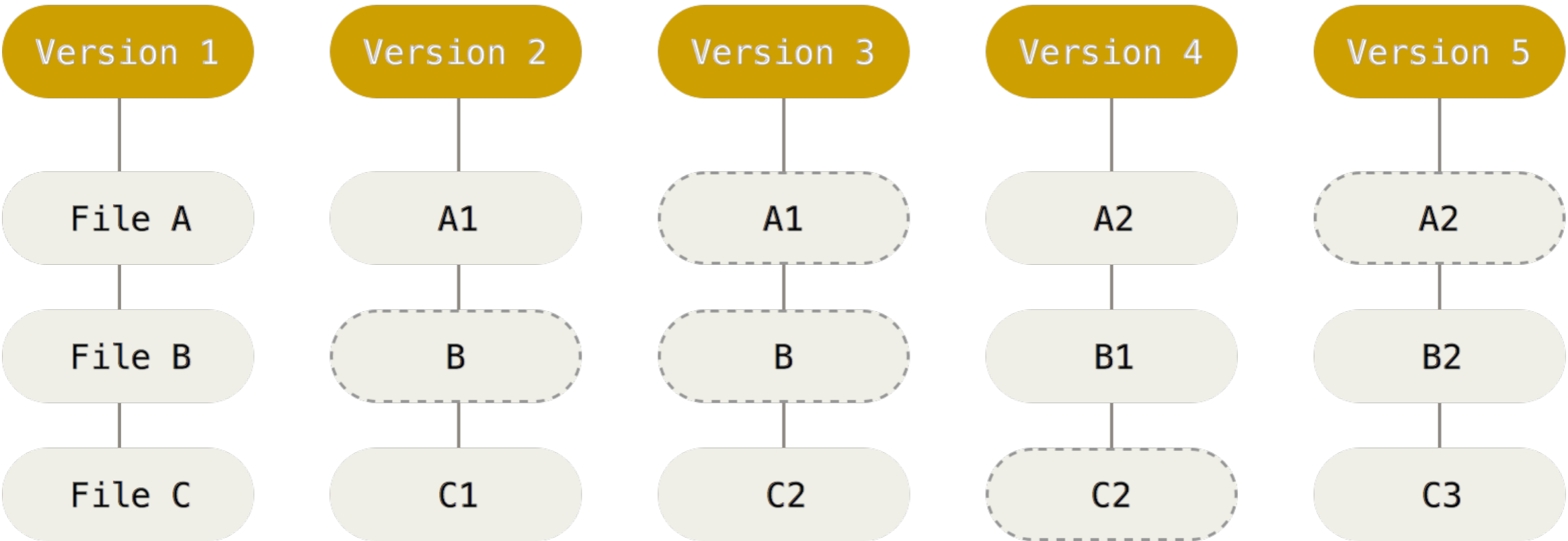
- Developed by Linus Torvalds for the Linux kernel in 2005
- Currently the most widely adopted version control system
- Goals
 - Support for non-linear development
 - Fully distributed
 - Efficient handling of large projects (like Linux)



Version control with Git: snapshots

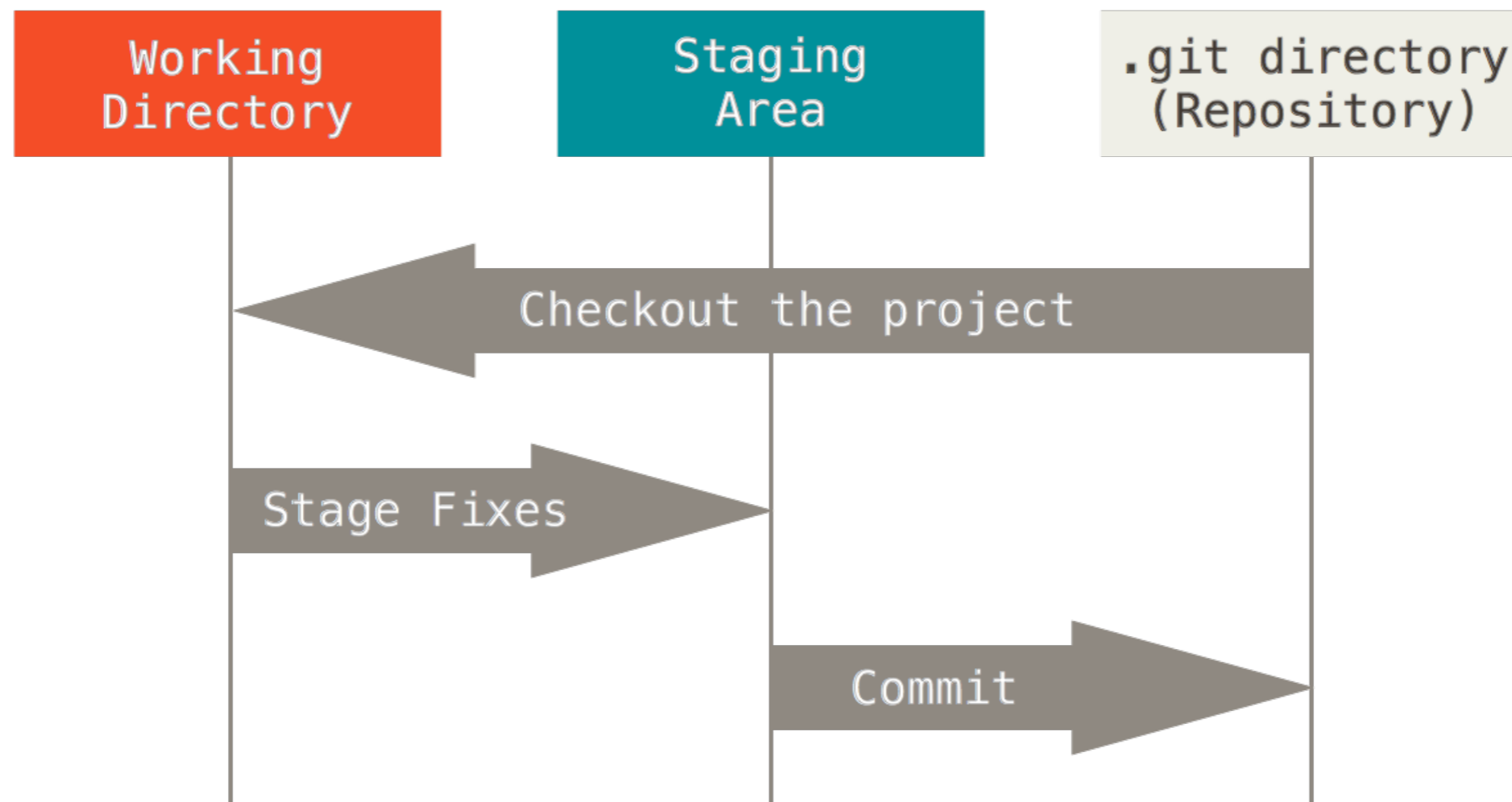
Stream of snapshots.

Checkins Over Time



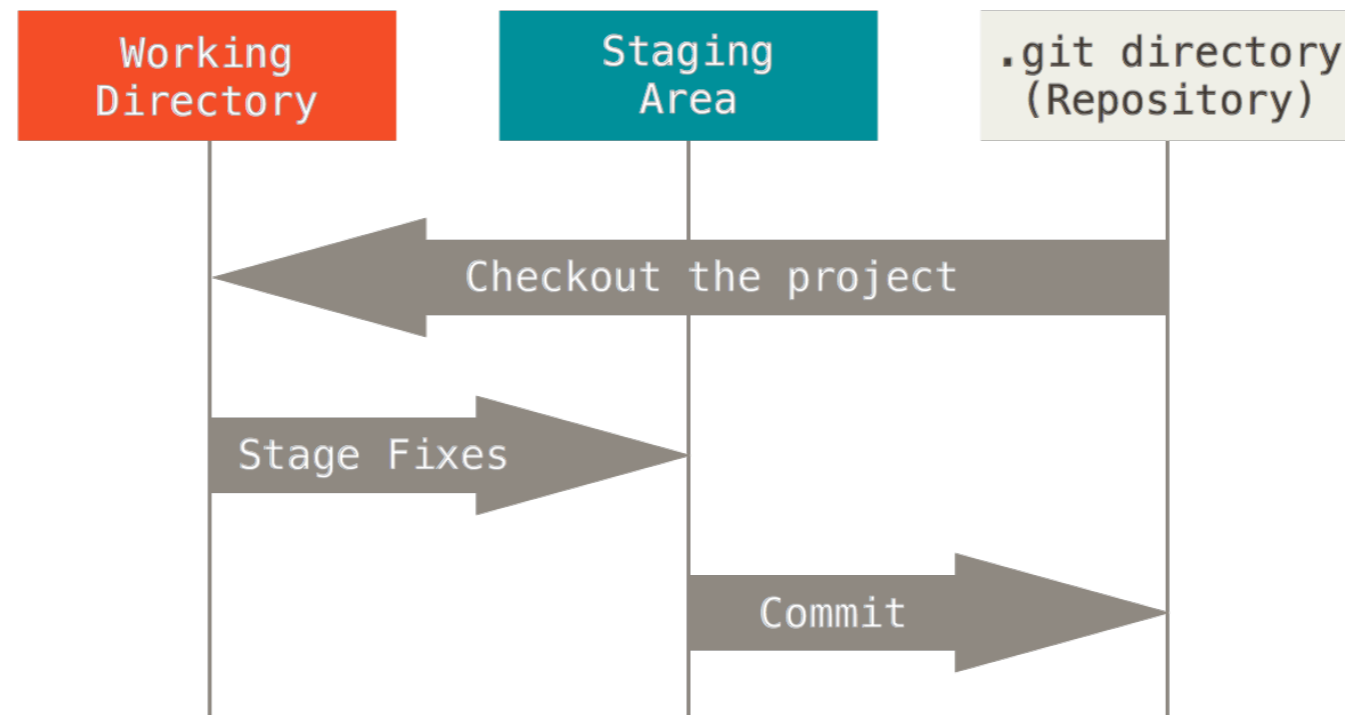
Version control with Git: states

Files are committed, modified, or staged.



Version control with Git: workflow

1. Modify files in the working directory.
2. Stage the files, adding snapshots of them to the staging area.
3. Do a commit, which takes the files as they are in the staging area and stores that snapshot permanently.




Working with Git: basic operations

- Setting up
- Creating or cloning a repository
- Staging and committing changes
- Pushing and pulling
- Branching and merging

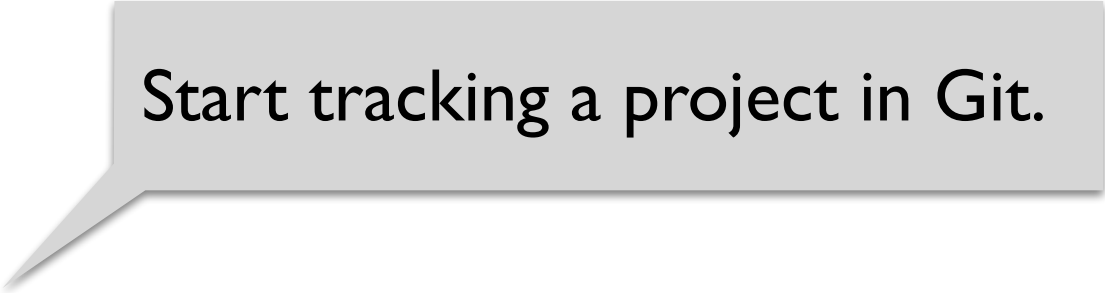
Working with Git: first-time setup

```
$ git config --global user.name "John Doe"  
$ git config --global user.email johndoe@example.com  
$ git config --global core.editor emacs
```



Set your identity and preferred editor.

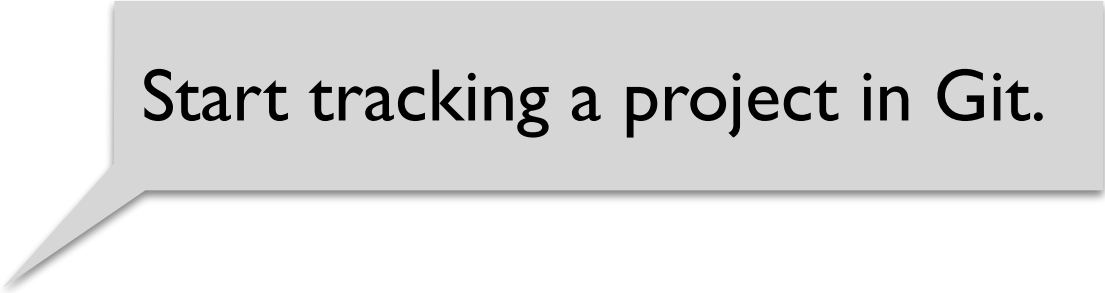
Working with Git: creating or cloning a repo



Start tracking a project in Git.

```
$ git init
```

Working with Git: creating or cloning a repo



Start tracking a project in Git.

```
$ git init
```

```
$ git clone https://github.com/emina/cse403
```

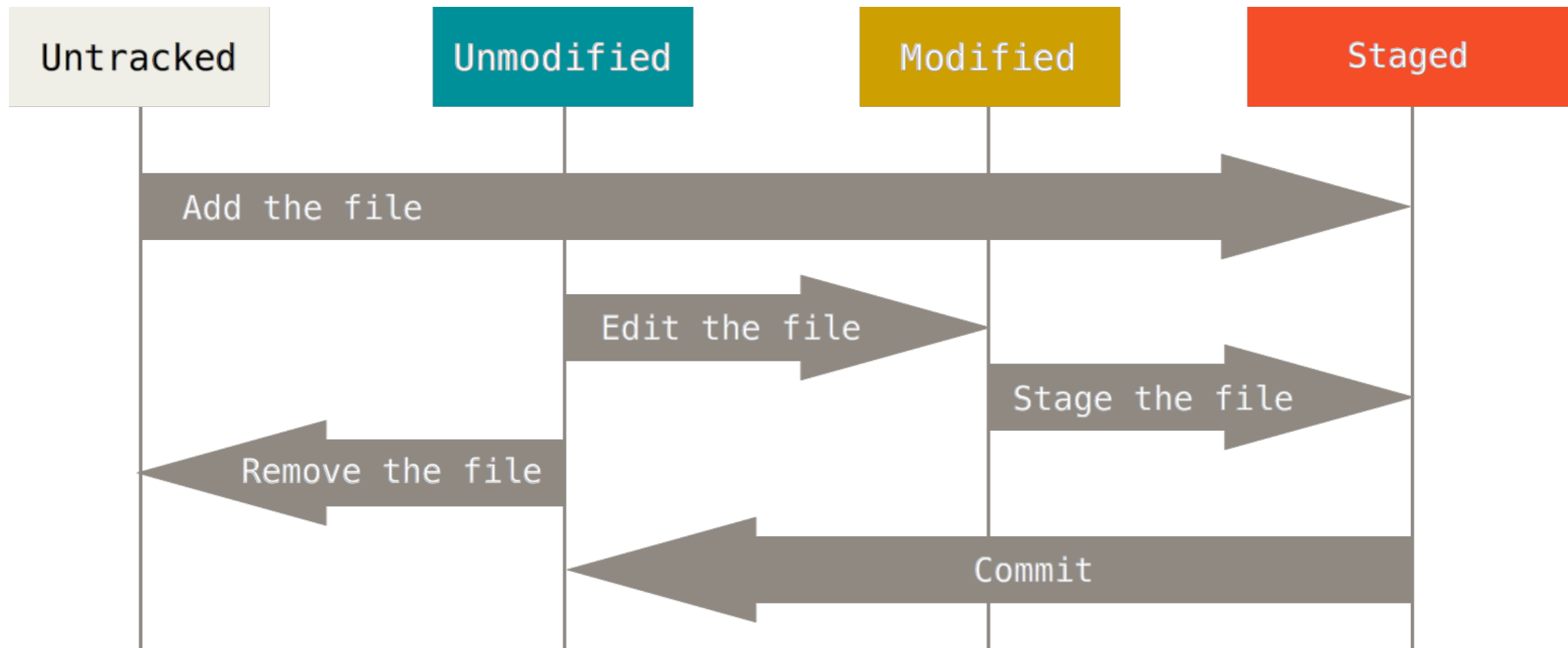


Clone an existing repository.

Working with Git: staging and committing

```
$ echo 'My Project' > README
```

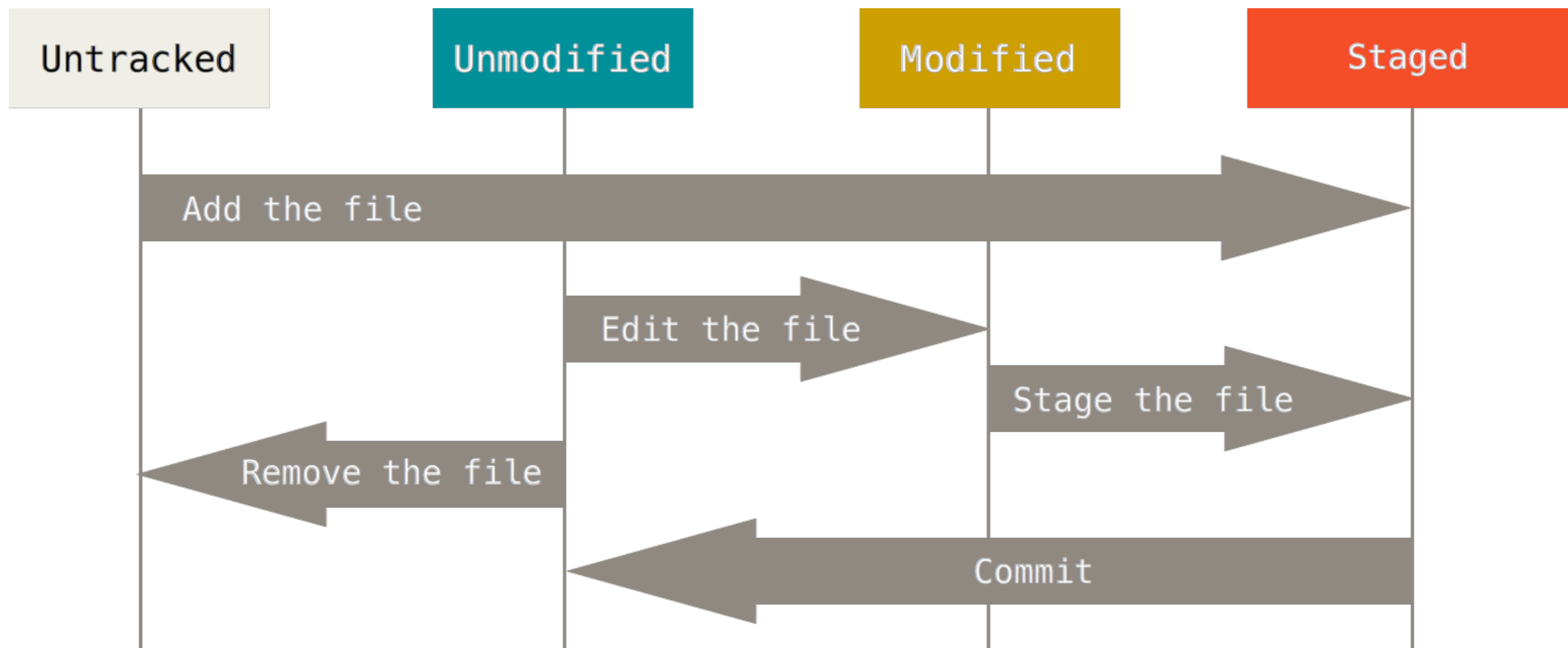
Untracked



Working with Git: staging and committing

```
$ echo 'My Project' > README  
$ git add README
```

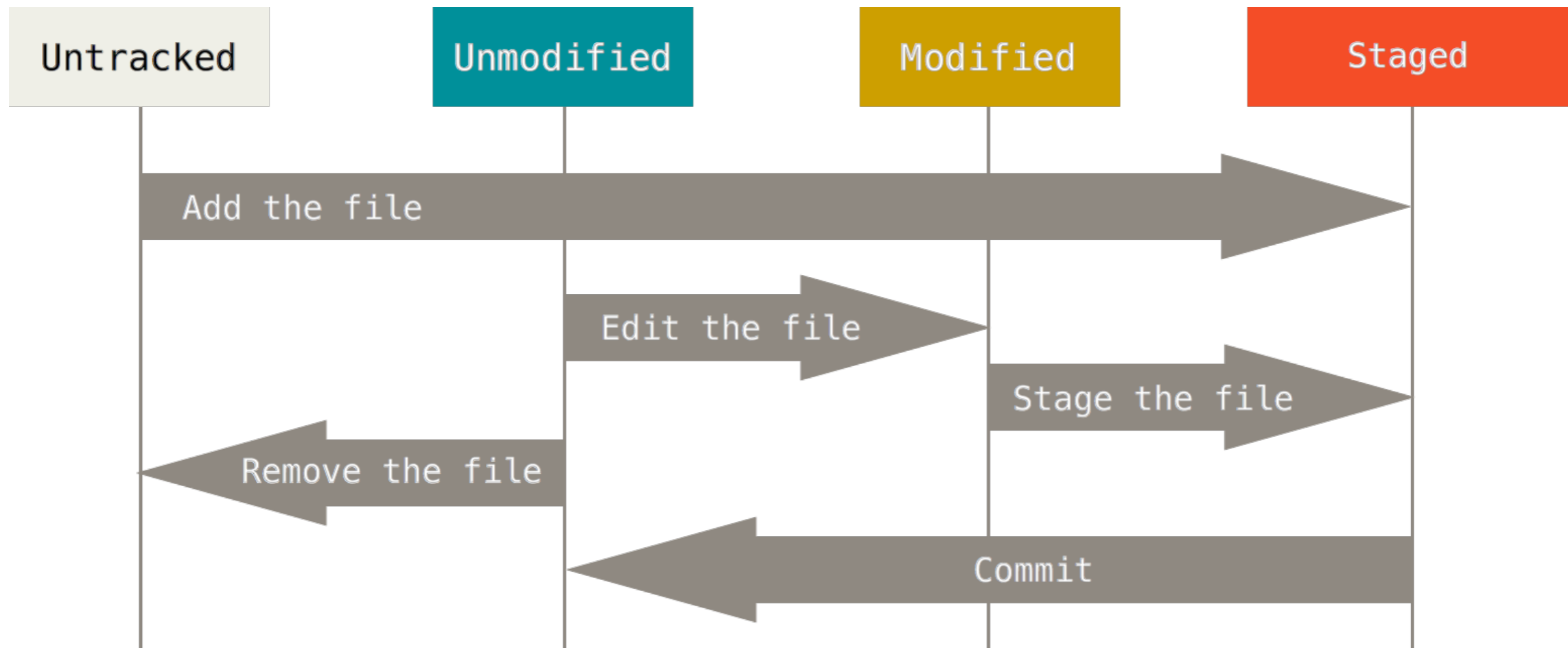
Staged



Working with Git: staging and committing

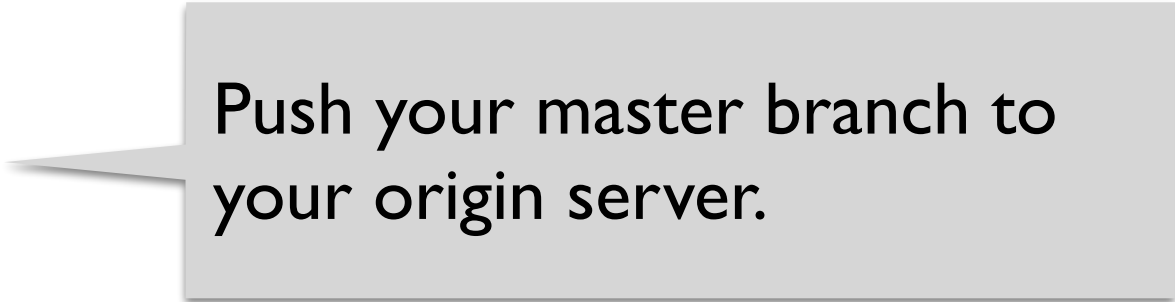
```
$ echo 'My Project' > README  
$ git add README  
$ git commit -m "readme"
```

Unmodified



Working with Git: pushing changes

```
$ git push origin master
```



Push your master branch to your origin server.

Working with Git: pushing changes

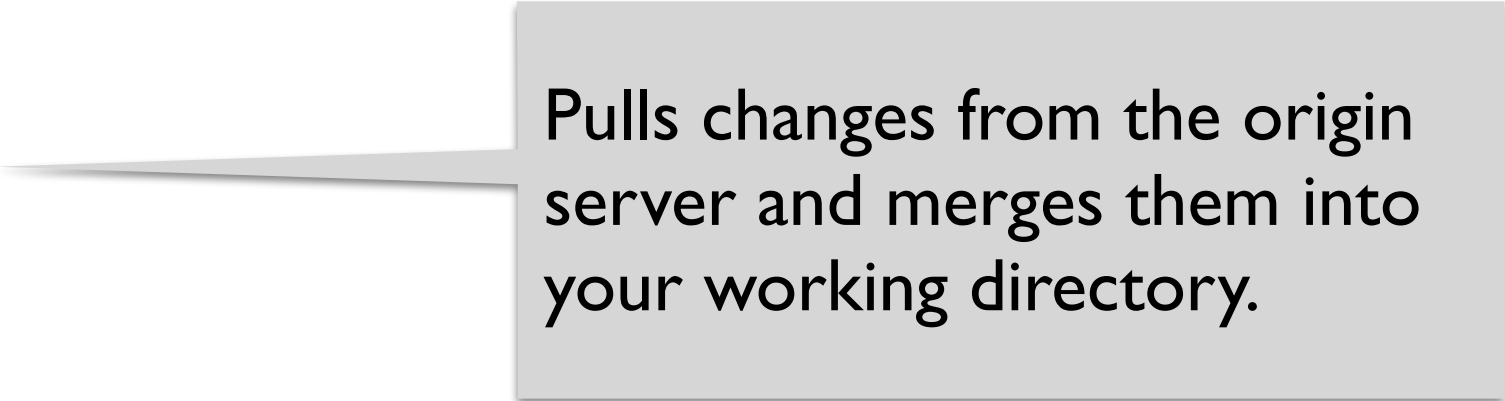
```
$ git push origin master
```

If out of sync with the origin, push will be rejected. Pull, merge, and try again.

Push your master branch to your origin server.

Working with Git: pulling changes

```
$ git pull
```



Pulls changes from the origin server and merges them into your working directory.

Working with Git: pulling changes

```
$ git pull
```

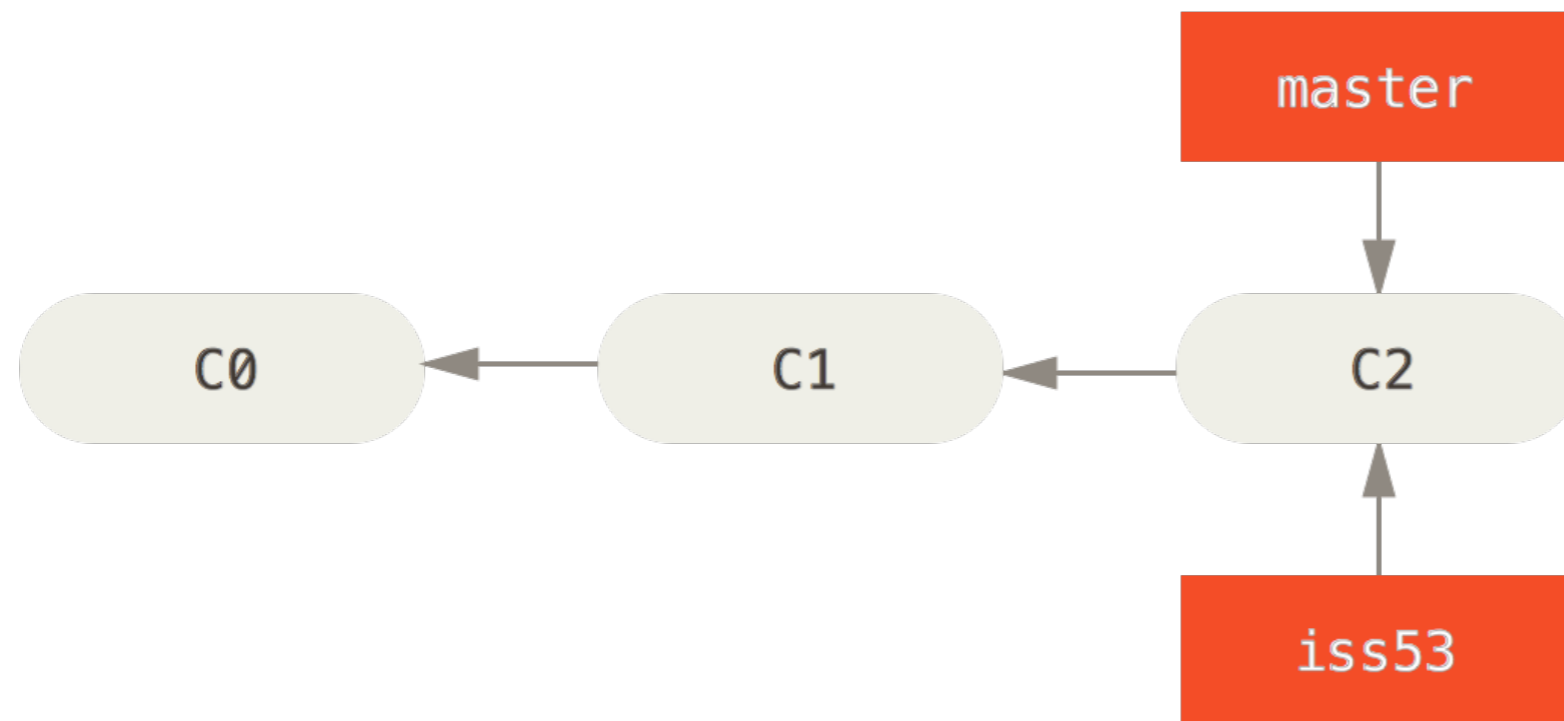
If there are conflicting changes, merge manually, add, and commit.

Pulls changes from the origin server and merges them into your working directory.

Working with Git: branching

Creates a branch and switches to it.

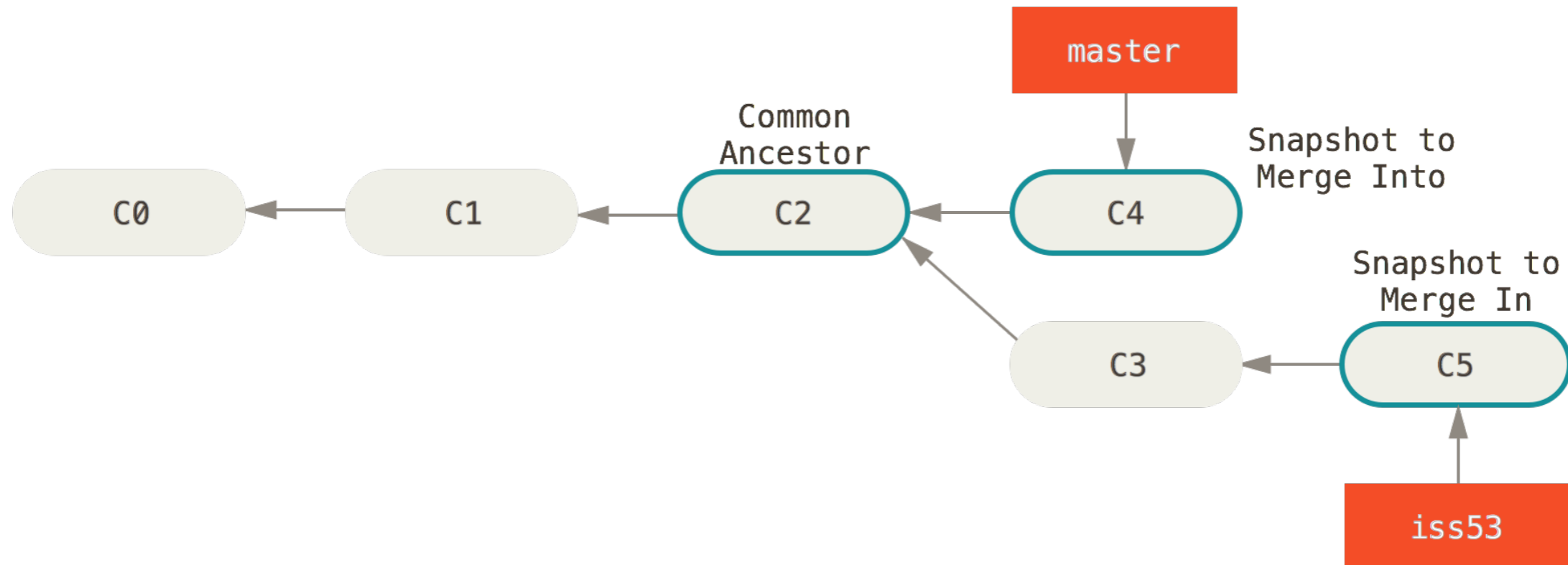
```
$ git checkout -b iss53
```



Working with Git: merging

```
$ git checkout master
```

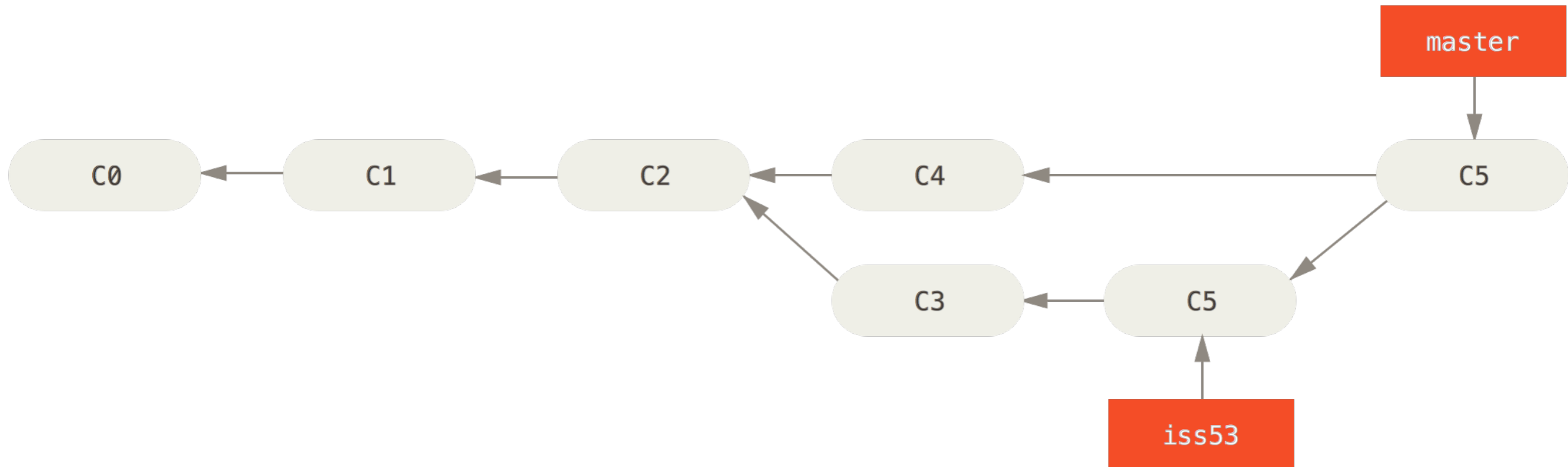
Switch to the master branch,
...



Working with Git: merging

```
$ git checkout master  
$ git merge iss53
```

Switch to the master branch,
and merge changes from iss53
into the master.



Hints for using Git: what not to commit

- Avoid binary files (especially simultaneous editing)
 - Word .doc files
- Do not commit generated files
 - Binaries (e.g., .class files), etc.
 - Wastes space in repository
 - Causes merge conflicts

Hints for using Git: commit often

- Make many small commits, not one big one
 - Easier to understand, review, merge, revert
- How to make many small commits:
 - Do only one task at a time and commit after each one
 - Create a new clone for each simultaneous task
 - Create a branch for each simultaneous task
 - Somewhat more efficient
 - Somewhat more complicated and error-prone
 - Easier to share unfinished work with teammates

Hints for using Git: synchronize often

- Pull often
 - Avoid getting behind the master or your teammates
- Push as often as practical
 - Don't destabilize the master build
 - Automatic testing on each push is a good idea

Hints for using Git: avoid merge conflicts

- Modularize your work
 - Divide work so that individuals or subteams “own” a module
 - Other team members only need to understand its specification
 - Requires good documentation and testing
- Communicate about changes that may conflict
 - But don't overwhelm the team in such messages

Summary

- Version control systems record changes to a file or set of files over time so that you can recall specific versions later.
- Git is the recommended VCS for 403.
- Learn more at <http://git-scm.com/book/en/>

