

# Git basics

Ángel de Vicente (*[angel.de.vicente@iac.es](mailto:angel.de.vicente@iac.es)*)

Date: November 13, 2020



# Index

1. Git basics
2. Git basic usage
3. Branches
4. Merging

# **1. Git basics**

## Basic concepts:

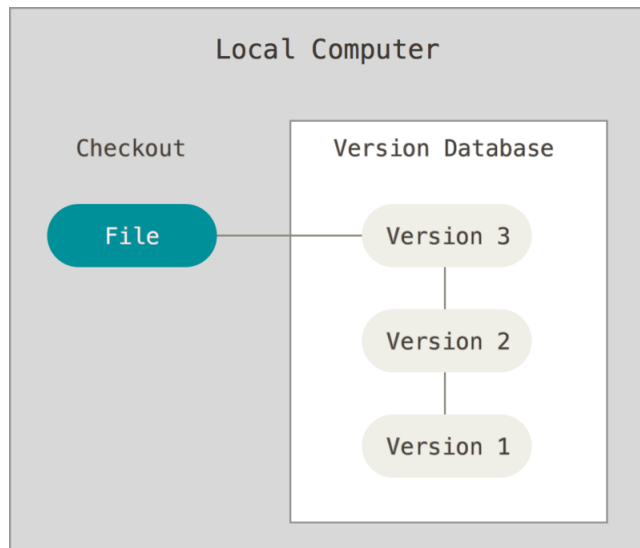
- What is version control?
- What do we version control? (text vs. binary files)
- Several version control systems (VCS): (*subversion*, *git*, *bazaar*, ...)
- You have probably already used a VCS directly or indirectly (e.g. Dropbox, Overleaf)
- Version control main concepts: *commit/checkout*, *branching*, *merging*
- Main benefits of version control systems: travelling back in time, reproducibility, collaboration

# Types of Version Control Systems

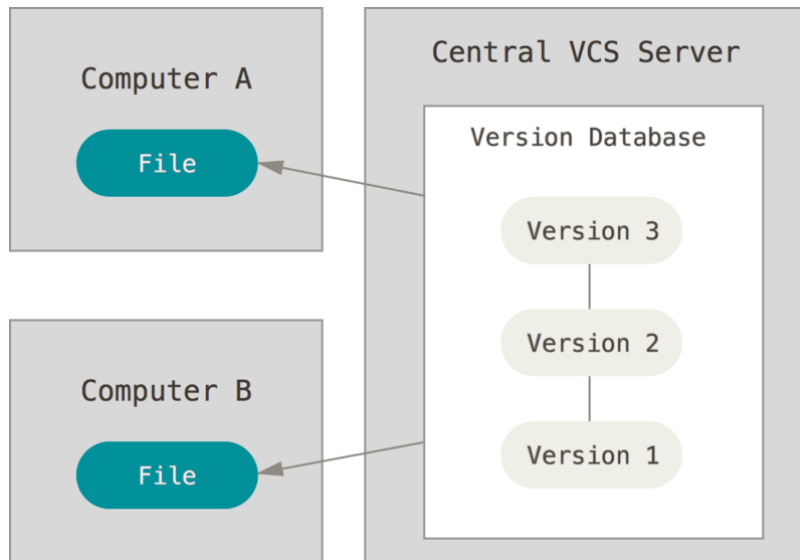
- *Manual VCS (don't be the one doing this!)*



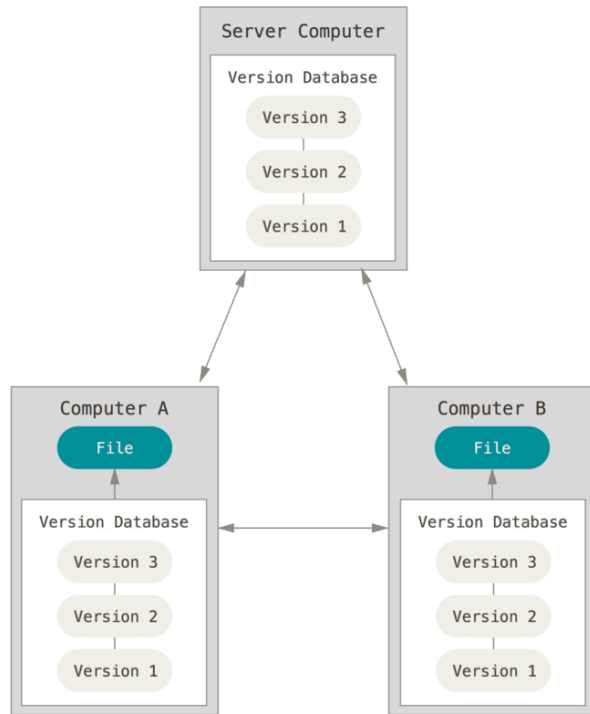
- Local VCS (e.g. RCS)



- Centralized VCS (e.g. Subversion)



- Distributed VCS (e.g. Git)





## Setting up git

In [3]: `! git --version`

```
git version 2.28.0
```

- *git config* (details at: <https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup> (<https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup>))
- Configuration variables stored in three different places:
  - */etc/gitconfig* file (*--system* option)
  - *~/.gitconfig* or *~/.config/git/config* file (*--global* option)
  - *config* file in the Git directory (*--local* option)

## Setting up git (2)

- Basic settings

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

- Many other options, e.g:

```
$ git config --global core.editor emacs  
$ git config --global color.ui "auto"
```

## Sample settings

In [7]: `! git config --list ## --show-origin`

```
user.email=angel.de.vicente@iac.es
user.name=Angel de Vicente
color.ui=true
color.status=auto
color.branch=auto
credential.helper=cache --timeout=28800
diff.jupyternotebook.command=git-nbdiffdriver diff
merge.jupyternotebook.driver=git-nbmergedriver merge %0 %A %B %L %P
merge.jupyternotebook.name=jupyter notebook merge driver
difftool.nbdime.cmd=git-nbdifftool diff "$LOCAL" "$REMOTE" "$BASE"
difftool.prompt=false
mergetool.nbdime.cmd=git-nbmergetool merge "$BASE" "$LOCAL" "$REMOTE" "$MERGE
D"
mergetool.prompt=false
pull.rebase=false
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=https://github.com/angel-devicente/git-workshop.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
remote.pushdefault=origin
branch.master.remote=origin
branch.master.merge=refs/heads/master
```

## Getting help

```
In [19]: ! git --help | head -n 18
```

```
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]
```

These are common Git commands used in various situations:

start a working area (see also: `git help tutorial`)

clone	Clone a repository into a new directory
init	Create an empty Git repository or reinitialize an existing one

work on the current change (see also: `git help everyday`)

add	Add file contents to the index
mv	Move or rename a file, a directory, or a symlink
restore	Restore working tree files
rm	Remove files from the working tree and from the index
sparse-checkout	Initialize and modify the sparse-checkout

```
In [4]: ! git status -h
```

```
usage: git status [<options>] [--] <pathspec>...
```

```
-v, --verbose          be verbose
-s, --short            show status concisely
-b, --branch           show branch information
--show-stash           show stash information
--ahead-behind         compute full ahead/behind values
--porcelain[=<version>]
                        machine-readable output
--long                show status in long format (default)
-z, --null             terminate entries with NUL
-u, --untracked-files[=<mode>]
                        show untracked files, optional modes: all, normal, no. (Default: all)
--ignored[=<mode>]     show ignored files, optional modes: traditional, matching, no. (Default: traditional)
--ignore-submodules[=<when>]
                        ignore changes to submodules, optional when: all, dirty, untracked. (Default: all)
--column[=<style>]     list untracked files in columns
--no-renames           do not detect renames
-M, --find-renames[=<n>]
                        detect renames, optionally set similarity index
```

```
In [19]: ! git status --help | head -n 22
```

GIT-STATUS(1)

Git Manual

GIT-STATUS(1)

#### NAME

git-status - Show the working tree status

#### SYNOPSIS

git status [<options>...] [--] [<pathspec>...]

#### DESCRIPTION

Displays paths that have differences between the index file and the current HEAD commit, paths that have differences between the working tree and the index file, and paths in the working tree that are not tracked by Git (and are not ignored by gitignore(5)). The first are what you would commit by running git commit; the second and third are what you could commit by running git add before running git commit.

#### OPTIONS

-s, --short

Give the output in the short-format.

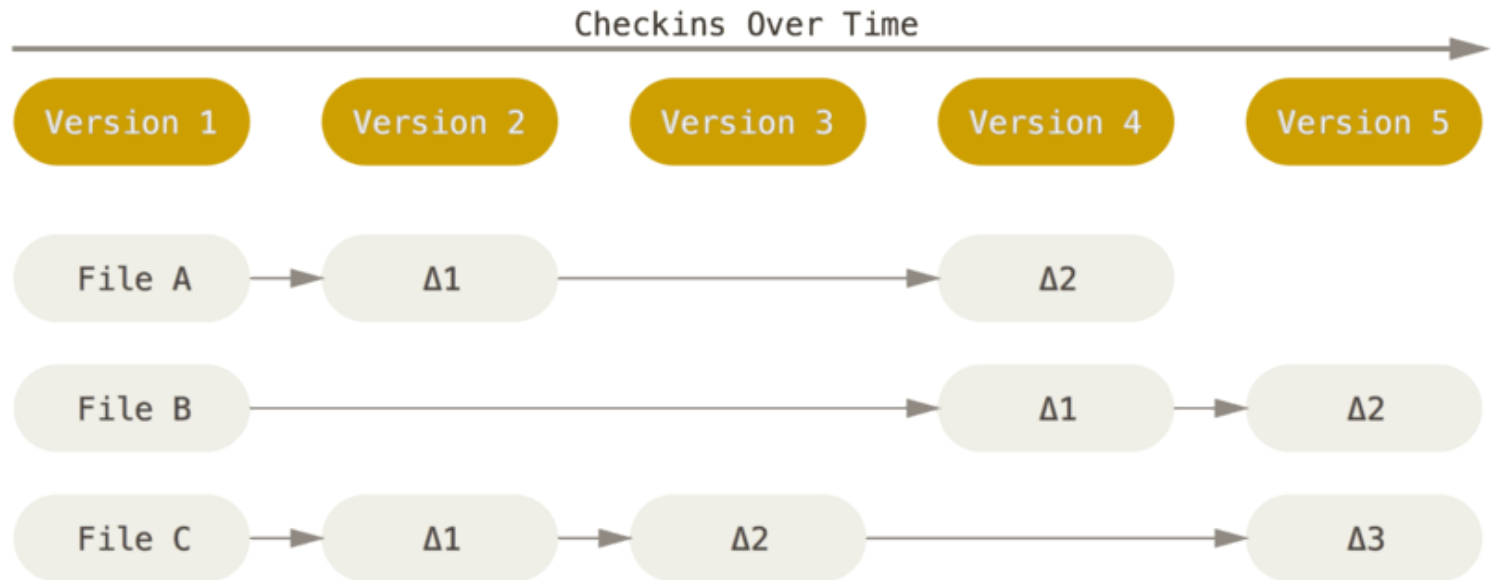
-b, --branch

Show the branch and tracking info even in short-format.

## Main ideas in Git

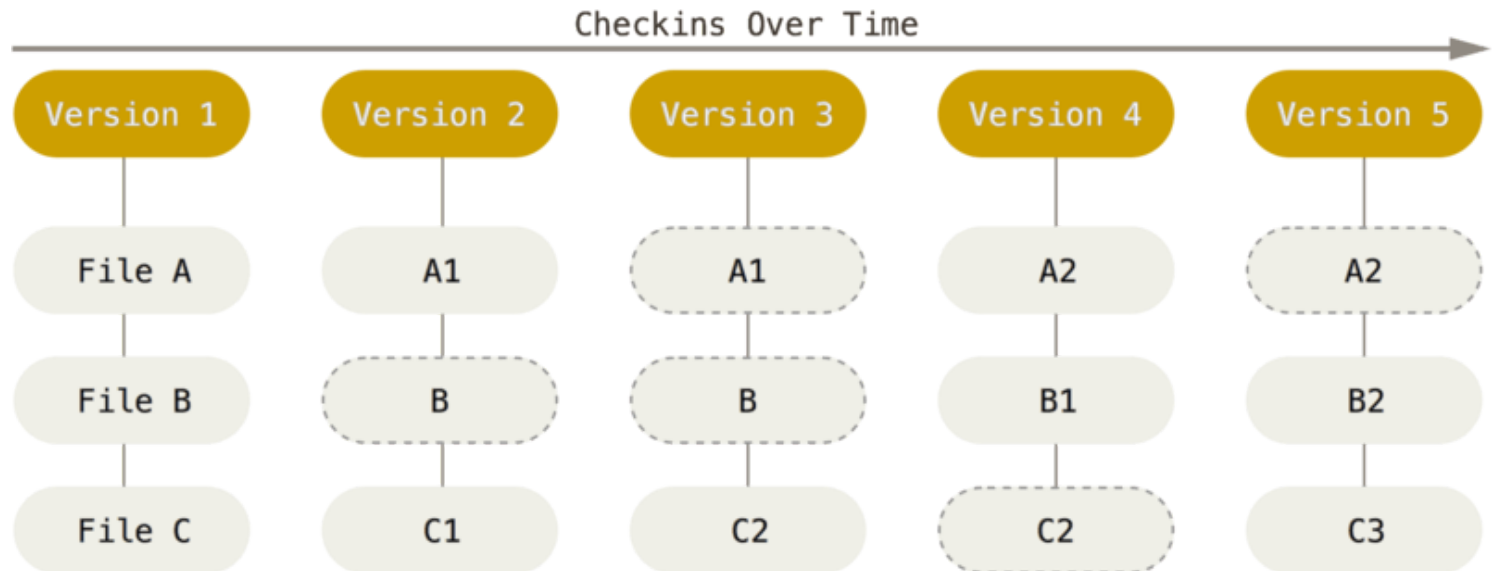
- Nearly all operations are local
  - `.git` directory keeps all history
  - very little that you cannot do when offline
- Integrity
  - Everything checksummed before stored
  - Hashes: `24b9da6552252987aa493b52f8696cd6d3b00373`
- Snapshots, not differences

- Delta-based VCS



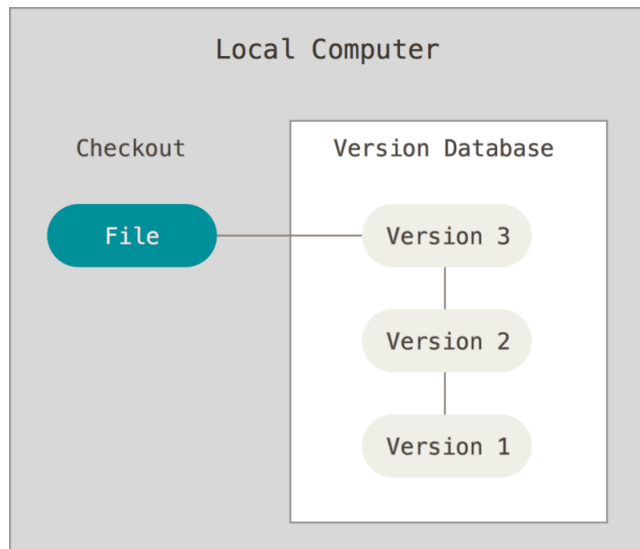


- Snapshots (Git)



## **2. Git basic usage**

## Using git as a local VCS



## Create a repository

### **git init**

In [15]:

```
%%bash  
  
cd ../Demos/Repos  
rm -rf First_Demo ; mkdir First_Demo ; cd First_Demo  
  
git init -q  
ls -al
```

```
total 12  
drwxr-xr-x 3 angelv angelv 4096 Oct 28 11:43 .  
drwxr-xr-x 6 angelv angelv 4096 Oct 28 11:43 ..  
drwxr-xr-x 7 angelv angelv 4096 Oct 28 11:43 .git
```

- All repository information goes to .git directory (DO NOT EDIT by hand)

In [16]:

```
%%bash
```

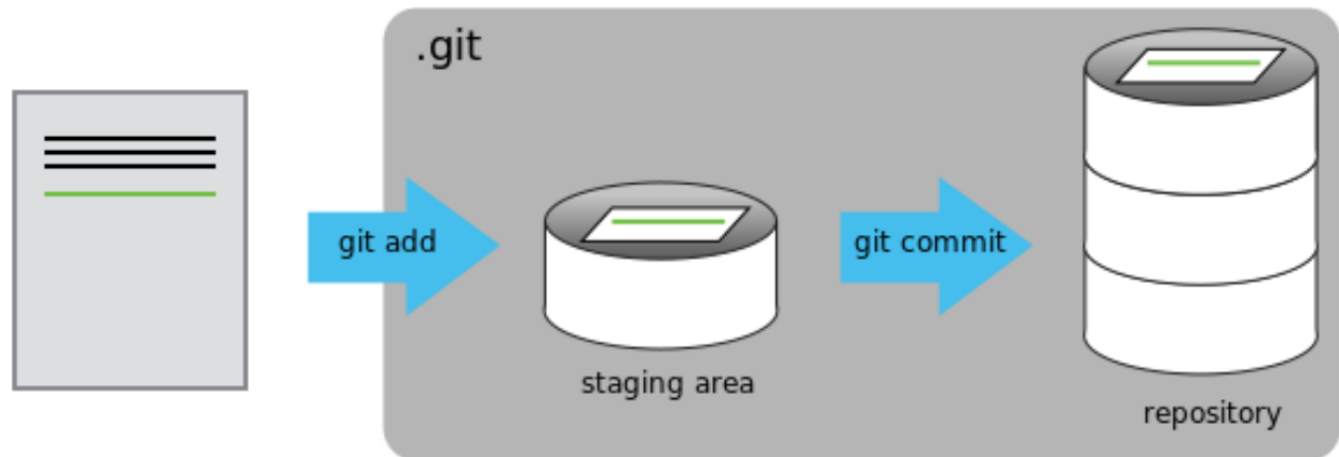
```
cd ../Demos/Repos  
ls First_Demo/.git
```

```
branches  
config  
description  
HEAD  
hooks  
info  
objects  
refs
```

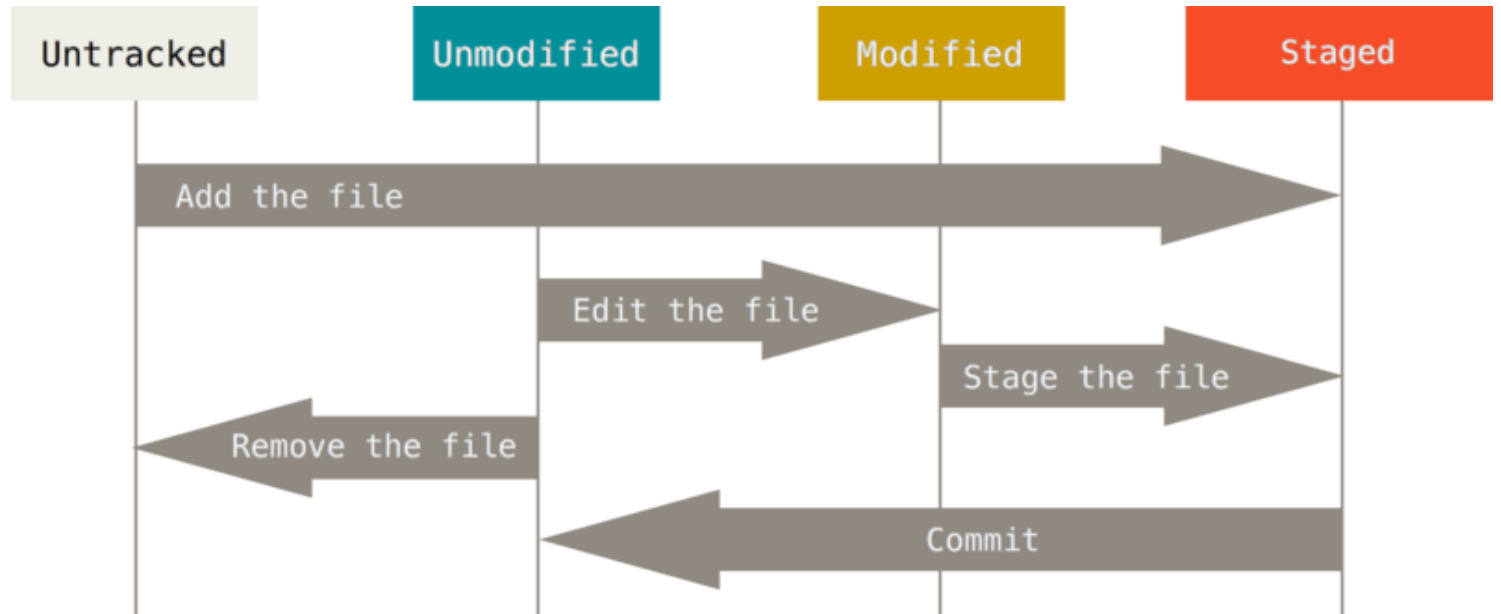
## Recording changes

- `git status`
- `git add`
- `git commit`
- `git rm`
- `git mv`

- Committing changes



- Tracking changes





## Recording changes demo

*demo-1-1.cast* (<https://asciinema.org/a/Rff5PnrBC79FMI7ThpbWBLX8D?autoplay=1&cols=180&rows=40>).

## Viewing history

- `git log`
- `git diff`

## Viewing history demo

[demo-1-2.cast \(https://asciinema.org/a/zHPylAmuKN3TWmGhQJiPOidZd?autoplay=1&cols=180&rows=40\)](https://asciinema.org/a/zHPylAmuKN3TWmGhQJiPOidZd?autoplay=1&cols=180&rows=40).

- Commit messages are important (*don't do this*)

```
ae69e17 * last
e6c4dc1 * last
6ee6e4d * last
0df6fcd * last
eacd473 * thesis
56b21d2 * last
c17ac01 * thesis LAST INDEEDgit statusgit statusgit status sent to referees this version
42e4c33 * last tt
e2b0b24 * last thesisgit statusgit status
e32e4e6 * tesis
aef739c * last
0481ed3 * last hopefully
961b606 * last version BEFORE last round
e04d68b * t
904d9e7 * crap thesis last
f1bcf71 * last
305a89a * th
e644f77 * th
fe20449 * th
93ada46 * added png
5bac6ff * cap 6
6f14455 * finalslopegit status
19a6ba7 * th
1431e0c * d
9cbebef * merge
      |\
54ec53b | * hh
3657531 * | th
      |/\
ff37a06 * th
650a99d * th
997d824 * mergmergee
      |\
e7f685d | * last
e5d1f8d * | last
      |/\
ec0ef72 * added app zdif
b8844e0 * th
3b8c792 * t
1-UUU:@%--F21 magit-log: finalpaper Top (1,0) (Magit Log ivy Projectile yas) 11:22 0.35 Mail
```

## Undoing things

- `git commit --amend`
- `git restore --staged CONTRIBUTING.md`
- `git restore README.md`

## Undoing things demo

[demo-1-3.cast](https://asciinema.org/a/Sq5xBDsbbbdJ1AxdZmz755xaJ?autoplay=1&cols=180&rows=40) (<https://asciinema.org/a/Sq5xBDsbbbdJ1AxdZmz755xaJ?autoplay=1&cols=180&rows=40>).

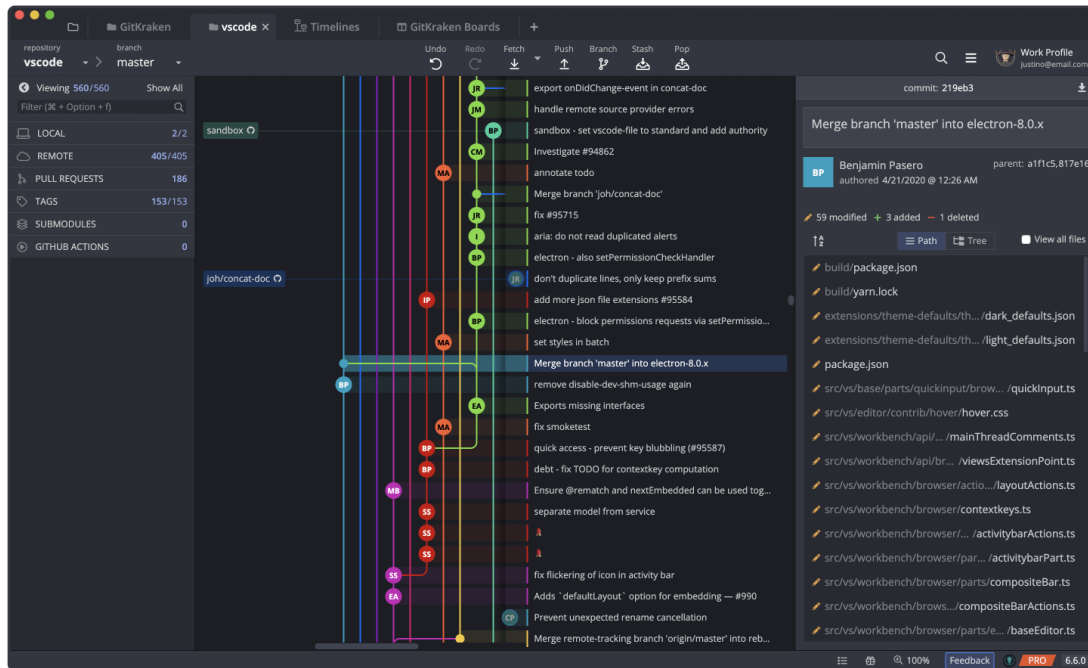
- In older versions of `git`, instead of `git restore` you would use:
  - `git reset HEAD ...`
  - and `git checkout ...`

(see examples in <https://git-scm.com/book/en/v2/Git-Basics-Undoing-Things> (<https://git-scm.com/book/en/v2/Git-Basics-Undoing-Things>))

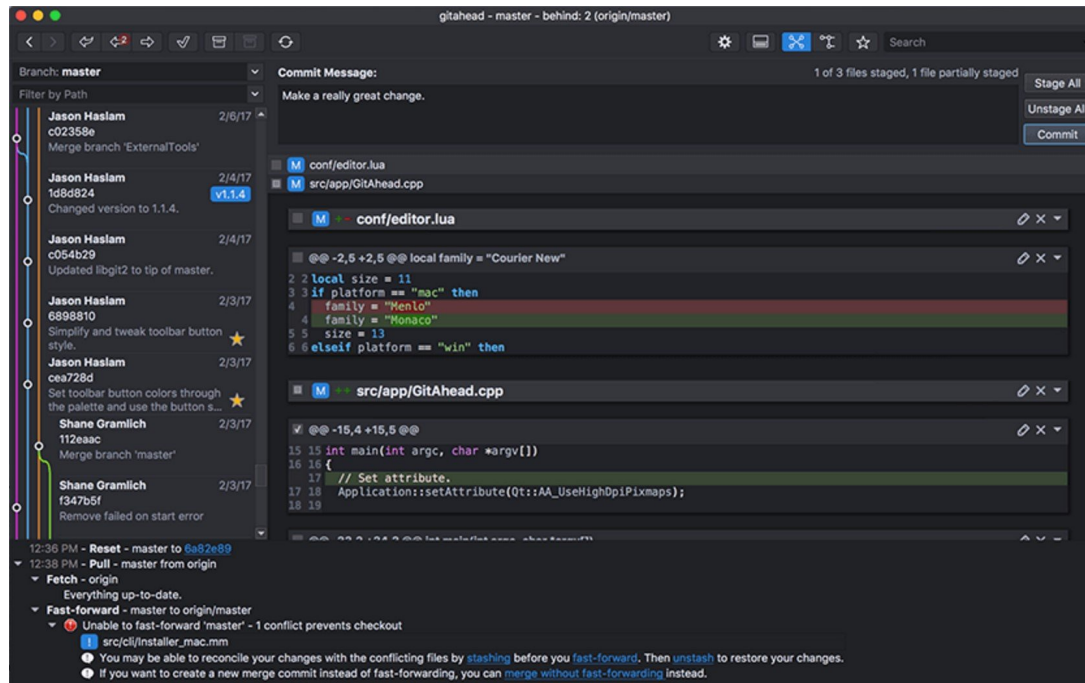
## Command-line vs GUIs

- Command-line lets you:
  - access *every* aspect of Git
  - can be automated in scripts
  - 'lingua franca'
- GUIs
  - the entry barrier for users is lower
  - can group common usage patterns
  - help with advanced options
  - choose your poison: GitKraken, GitAhead, Magit, ...
    - <https://git-scm.com/download/gui/linux> (<https://git-scm.com/download/gui/linux>)

- GitKraken <https://www.gitkraken.com/> (<https://www.gitkraken.com/>)



- GitAhead <https://gitahead.github.io/gitahead.com/>  
(<https://gitahead.github.io/gitahead.com/>)





- Magit (for Emacs) <https://magit.vc/> (<https://magit.vc/>).

```

0888d9c * master origin/master Merge pull request #257 from jotoeri/add-guilink Jessica Lord 1 year
7934d89 \ Adding a link to the GUI overview jotoeri 1 year
ae749cd \ Merge pull request #252 from 0x15f9/master Jessica Lord 1 year
fb399c4 \ Fixed slight typo 0x15f9 1 year
8c64b82 \ Merge pull request #242 from franciosi/patch-2 Jessica Lord 1 year
9eba724 \ Update README.md Franciosi 1 year
aac8582 \ Merge pull request #238 from juananruiz/patch-1 Jessica Lord 2 years
67d5dee \ Cambiando "and" por "y" Juan Antonio Ruiz 2 years
adec755 \ Asegurate - Asegurate Jessica Lord 2 years
8a2594f \ Merge pull request #234 from diegocasillas/spanish-translati Jessica Lord 2 years
8d353b9 \ Fix typos in 11_merge_tada.html diegocasillas 2 years
3de27cb \ Fix typos in 10_requesting_you_pull_please.html diegocasillas 2 years
163256b \ Fix typos in 9_pull_never_out_of_date.html diegocasillas 2 years
996d499 \ Fix typos in 7_branches_arent_just_for_birds.html diegocasillas 2 years
788b85a \ Fix typos in 6_forks_and_clones.html diegocasillas 2 years
8c4d891 \ Fix typos in 5_remote_control.html diegocasillas 2 years
bdc4cbe \ Fix typos in 4_githubbin.html diegocasillas 2 years
8638829 \ 4.4.0 Merge pull request #231 from mberasategi/es_ES Jessica Lord 2 years
e1c59e3 \ Add es_ES to README mberasategi 2 years
9f5145c \ Set es_ES for es alias mberasategi 2 years
a78316a \ Add new locale to menus mberasategi 2 years
803664e \ Translate pages mberasategi 2 years
2a1ca1e \ Translate buttons mberasategi 2 years
686c2a4 \ Translate 11_merge mberasategi 2 years
1679c8d \ Translate 10_requesting_pullrequest mberasategi 2 years
94d5b8c \ Translate 9_pull mberasategi 2 years
b0c799c \ Translate 8_small_world mberasategi 2 years
88e8310 \ Translate 7_branches mberasategi 2 years
e159835 \ Translate 6_forks_clones mberasategi 2 years
18f5890 \ Translate 5_remote_control mberasategi 2 years
95c4bae \ Translate 4_githubbin mberasategi 2 years
378b180 \ Translate 3_commit_to_it mberasategi 2 years
86822d7 \ Change hello-world to hola-mundo mberasategi 2 years
a981cbe \ Change Git Shell to Git CMD for Windows mberasategi 2 years
8ff7237 \ Begin translating 3_commit-to-it Miren Berasategi 2 years
e189274 \ Translate 2_repository Miren Berasategi 2 years
8d9a210 \ Translate 1_get_git Miren Berasategi 2 years
c14968d \ Duplicate es_CO to jumpstart es_ES Miren Berasategi 2 years
7eaa55c \ Add list of supported languages to readme Jessica Lord 2 years
573d880 \ 4.3.0 version 4.3.0 Jessica Lord 2 years
91f9070 \ make lang button wider to fit language names Jessica Lord 2 years
2f6821 \ Merge pull request #217 from node-co/feature/spanish-translati Jessica Lord 2 years
1-000:0%--F21 magit-log: git-it-electron Top (5.0) (Magit Log ivy Projectile yas) 06:34 0.61 -----
98-000:0%--F21 magit: git-it-electron All (6.0) (Magit ivy Projectile yas) 06:34 0.61 -----
Author: Jessica Lord <jlord@glitch.com>
AuthDate: Sun Mar 10 20:49:34 2019 -0400
Commit: GitHub <noreply@github.com>
CommitDate: Sun Mar 10 20:49:34 2019 -0400
Parent: 8c64b82 Merge pull request #242 from franciosi/patch-2
Parent: fb399c4 Fixed slight typo
Contained: master
Follows: 4.4.0 (15)
Merge pull request #252 from 0x15f9/master
Fixed slight typo
1 file changed, 2 insertions(+), 2 deletions(-)
resources/contents/en-US/challenges/4_githubbin.html | 4 +++
[back]
09-000:0%--F21 magit-revision: git-it-electron All (3.0) (Magit Rev ivy Projectile yas) 06:34 0.61 -----
0:gnus# 1:diary# 2:org# 3:julia# 4:IPython# 5:IDL# 6:PORTA# 7:CORONA# 8:Git Workshop# 9:-# 23.566% 2020-05-06 06:34:51

```

### **3. Branches**

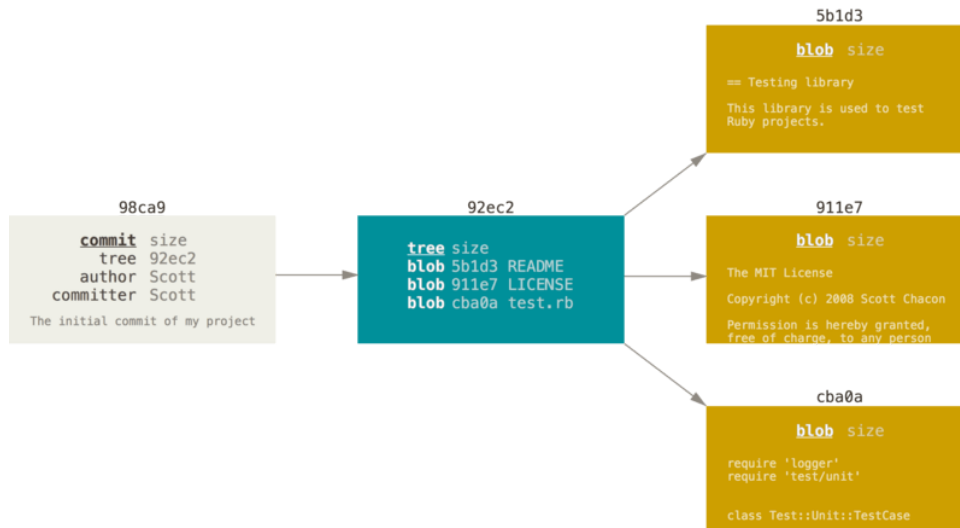
## Branches

- A lot of the power (and some confusion) in Git comes from branches
- Branches are the *killer* feature of Git, very lightweight compared to other VCS
- This encourages workflows that create branches and merge them very often.

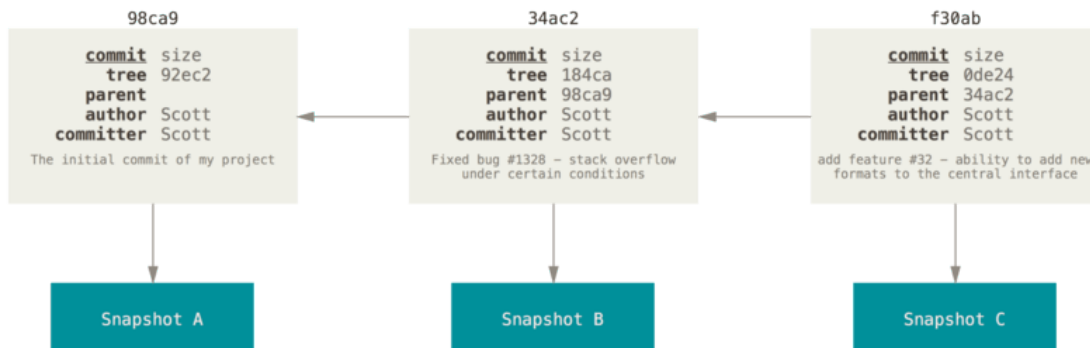
*So, let's try and see some of the Git internals to understand what branches are...*

(see <https://git-scm.com/book/en/v2/Git-Branching-Branched-Branches-in-a-Nutshell> (<https://git-scm.com/book/en/v2/Git-Branching-Branched-Branches-in-a-Nutshell>))

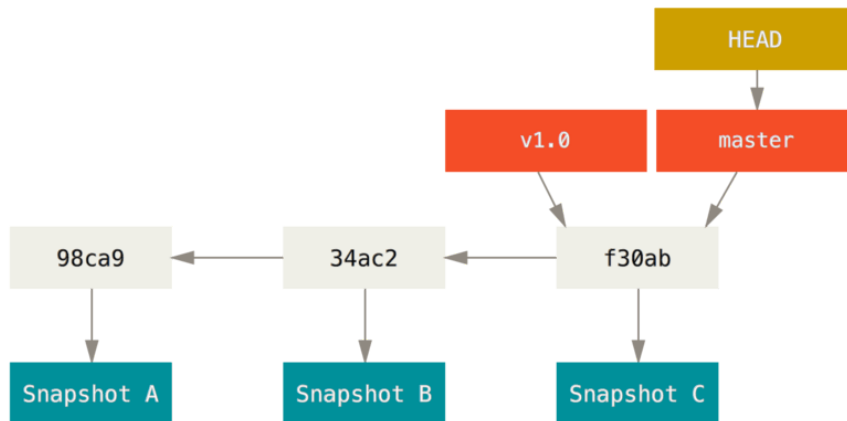
- A *commit* points to a *tree* object, and this to *blobs*
  - For us the important part is just the *commit*, which points to a *snapshot* of our work



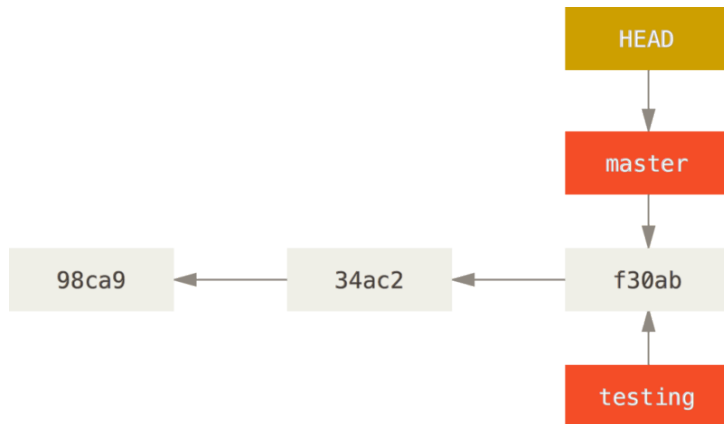
- Creating more *commits*, keeps a linked list
  - (which is basically what you see when you do `git log`)



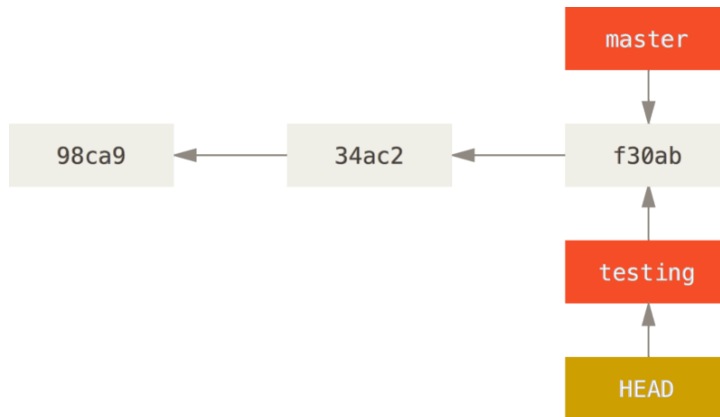
- A *branch* is just a pointer to one of these *commits*
  - *master* is created when you do `git init`, but it is no special
  - *HEAD* points to the branch we are working on



- Creating a branch
  - If we are working in *master* and we do `git branch testing`
    - *testing* branch is created, but we are still working on *master*
    - so you can see *HEAD* is still pointing to *master*

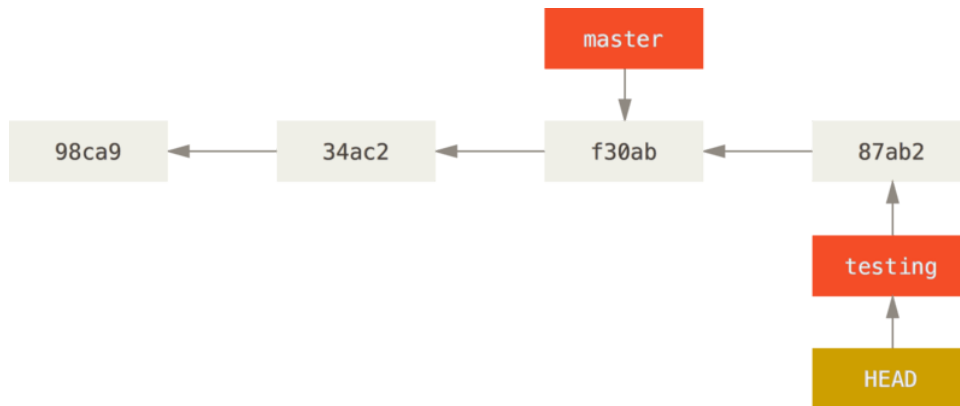


- To work with another branch
  - `git checkout testing` makes *HEAD* point to testing
    - when you switch branches in Git:
      - **files in your working directory will change.**
      - **if Git cannot do it cleanly, it will not let you switch at all.**

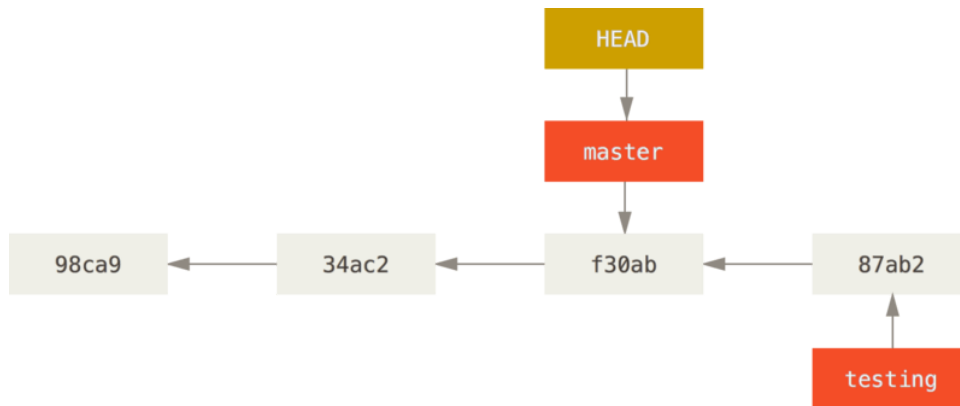




- Committing to *testing*
  - makes *testing* branch move forward
  - but *master* branch still points to the same commit



- Changing branches
  - `git checkout master` (if *working tree* is *dirty* we won't be able to change branches)



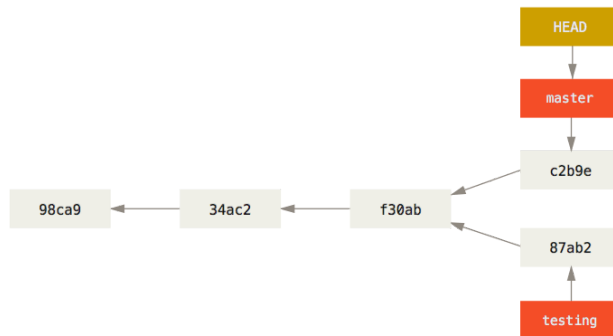
## Branching demo

- In *git-it-cp1*:
  - create a new branch *test\_branch1* and change to it
  - add a line to a file and commit changes
  - change back to branch *master* and verify the added line is not present in this branch
  - use `git log` to view the history of all branches (should be able to see in first position both branches: *master* and *test\_branch1*, plus *HEAD*)

[demo-1-5.cast \(https://asciinema.org/a/0du1PPZviaW6EoQ7DSATEiAqe?autoplay=1&cols=180&rows=40\)](https://asciinema.org/a/0du1PPZviaW6EoQ7DSATEiAqe?autoplay=1&cols=180&rows=40)

## The problem: divergent branches (applies to local and remote branches)

- Earlier we created a *testing* branch and committed some changes.
- Later we went back to the *master* branch.
- At that point, if you commit to master, you end up with divergent branches.
- If you want to incorporate the changes in *testing* to *master*, you might have conflicts (maybe both commits updated the same function).



## Two possible solutions to divergent branches

### **git merge**

Incorporates changes from the named commits (since the time their histories diverged from the current branch) into the current branch.

### **git rebase** (*look for "Intermediate/Advanced Git"*)

Apply all changes made in the current branch on top of another branch. (*This is a more advanced command, and you have to be careful with it*)

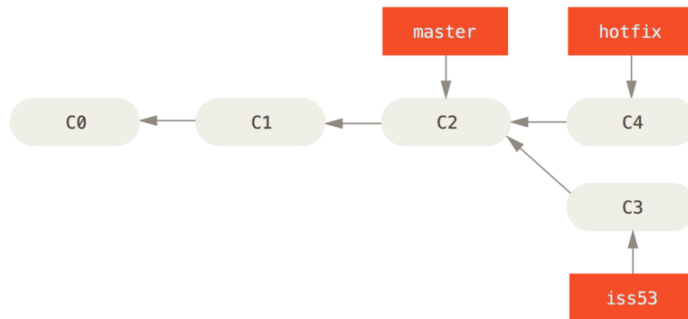
## 4. Merging

## Merging

- Fast-forward merges
  - the simplest
- No-conflict merges
  - very simple, and very common if working on your own
- Merges with conflicts
  - very usual when collaborating with others, specially if long time between commits

## Fast-forward "merge"

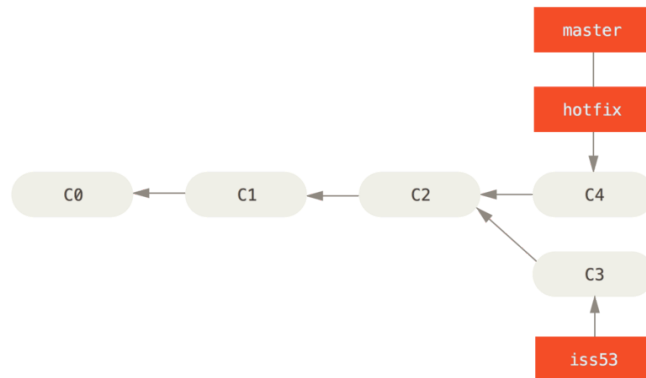
- Imagine you are in a situation like this, where:
  - you have two branches, started off the *master* branch, where you made one commit to each.
  - you want now to incorporate to *master* the changes done in branch *hotfix*





## Fast-forward "merge" (2)

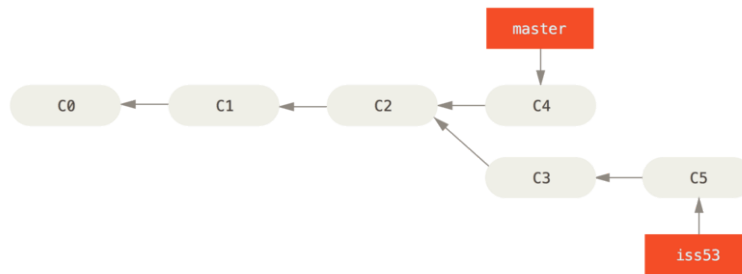
- Check out the master branch and merge the hotfix branch
  - This is a fast-forward merge (i.e. nothing really to merge, not divergent branch)



[demo-1-7.cast \(https://asciinema.org/a/ZbFhHkGvValGWF3HiF9ZiSyUG?autoplay=1&cols=180&rows=40\)](https://asciinema.org/a/ZbFhHkGvValGWF3HiF9ZiSyUG?autoplay=1&cols=180&rows=40)

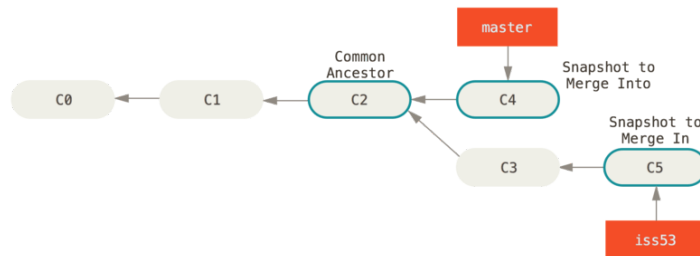
## No-conflicts merge

- Following from the previous situation:
  - you now have a branch *iss53* that has diverged from the *master* branch
  - a "fast-forward" is not possible when incorporating those changes to *master*



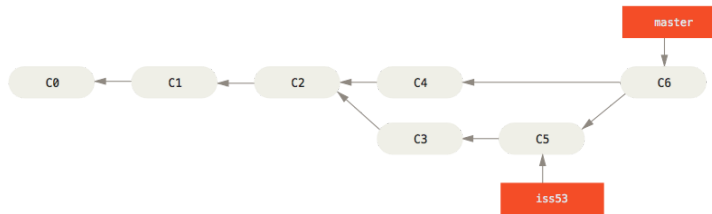
## No-conflicts merge (2)

- A `git merge` command will have to work harder now to find:
  - which commit is common to both lines
  - what changes were made in each branch so as to collect all changes
  - if changes in, e.g. different files/functions, merge possible w.o. conflicts



## No-conflicts merge (3)

- And we end up with a new "merge" commit in *master*:



**Note:** remember that `git merge iss53` means: merge branch *iss53* to my current branch (i.e. you want to issue the merge command with the destination branch checked-out)

[demo-1-8.cast \(https://asciinema.org/a/4SI3zhP3K0ds9I5SdmHXUe8d?autoplay=1&cols=180&rows=40\)](https://asciinema.org/a/4SI3zhP3K0ds9I5SdmHXUe8d?autoplay=1&cols=180&rows=40)

## Merge with conflicts

**When git cannot merge automatically, it will tell you:**

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

**The conflicting files will have conflict-resolution markers:**

```
<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">contact us at support@github.com</div>
>>>>>> iss53:index.html
```

## Merge with conflicts (2)

- To resolve the conflict by hand:
  - just edit the file with your usual text editor, leaving the correct resolution (and no markers)
  - run `git commit` when all conflicts have been resolved

[demo-1-9.cast](#)

<https://asciinema.org/a/MxjPsupBk3toHCQsOuEQBKbMF?autoplay=1&cols=180&rows=40>

- But you should really learn how to use an external tool (peek into "Intermediate Git")
  - `git mergetool` will tell you options and how to configure git
  - GitAhead demo: <https://www.youtube.com/watch?v=W-FHwUwE84M>  
<https://www.youtube.com/watch?v=W-FHwUwE84M>
  - Emacs + Magit + Ediff demo: [https://youtu.be/S86xsx\\_NzHc](https://youtu.be/S86xsx_NzHc)  
[https://youtu.be/S86xsx\\_NzHc](https://youtu.be/S86xsx_NzHc)

## References

- [Git main page \(https://git-scm.com/\)](https://git-scm.com/).
- [Pro Git book \(https://git-scm.com/book/en/v2\)](https://git-scm.com/book/en/v2).
- [Git cheatsheet \(https://ndpsoftware.com/git-cheatsheet.html\)](https://ndpsoftware.com/git-cheatsheet.html).