

Git basics

Ángel de Vicente (*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:

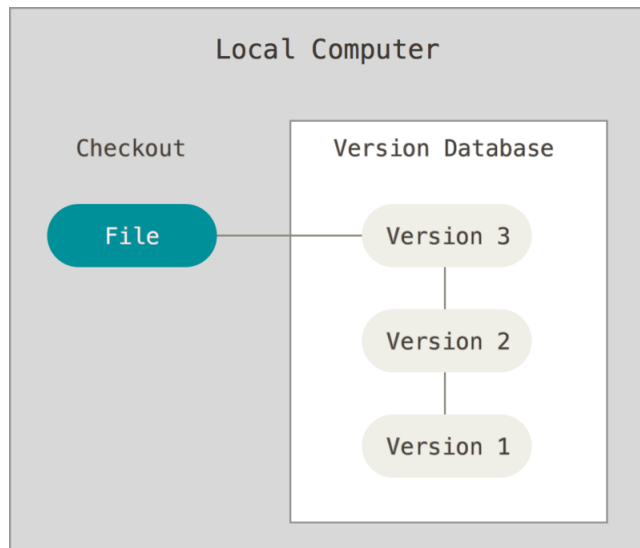
- What is version control?
- What do we version control? (text vs. binary files)
- Several version control systems (VCS): (*subversion*, *git*, *bazaar*, ...)
- You have probably already used a VCS directly or indirectly (e.g. Dropbox, Overleaf)
- Version control main concepts: *commit/checkout*, *branching*,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

Types of Version Control Systems

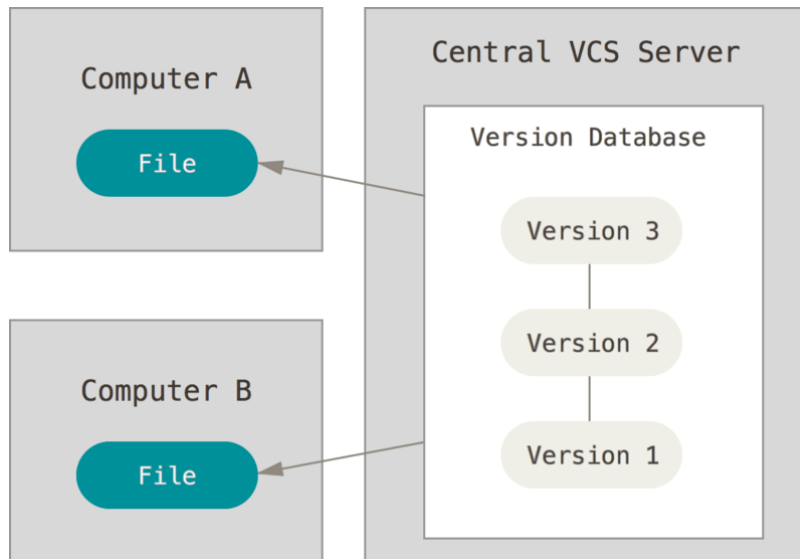
- *Manual VCS (don't be the one doing this!)*



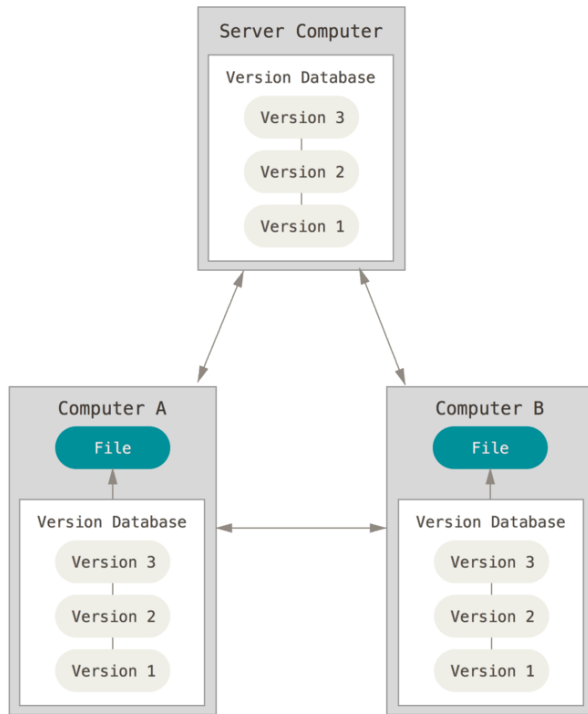
- Local VCS (e.g. RCS)



- Centralized VCS (e.g. Subversion)



- Distributed VCS (e.g. Git)



Setting up git

```
In [1]: %%bash
        ## %%bash not part of git. Just to be able to execute bash commands live during
        the presentation

        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 [2]: %%bash

```
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 [3]: %%bash
## | head -n 18  to print only the first 18 lines

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]: %%bash

```
git add -h | head -n 18
```

usage: git add [<options>] [--] <pathspec>...

-n, --dry-run	dry run
-v, --verbose	be verbose
-i, --interactive	interactive picking
-p, --patch	select hunks interactively
-e, --edit	edit current diff and apply
-f, --force	allow adding otherwise ignored files
-u, --update	update tracked files
--renormalize	renormalize EOL of tracked files (implies -u)
-N, --intent-to-add	record only the fact that the path will be added later
-A, --all	add changes from all tracked and untracked files
--ignore-removal	ignore paths removed in the working tree (same as --no-all)
--refresh	don't add, only refresh the index
--ignore-errors	just skip files which cannot be added because of errors
--ignore-missing	check if - even missing - files are ignored in dry run
--chmod (+ -)x	override the executable bit of the listed files

In [5]: %%bash

```
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.

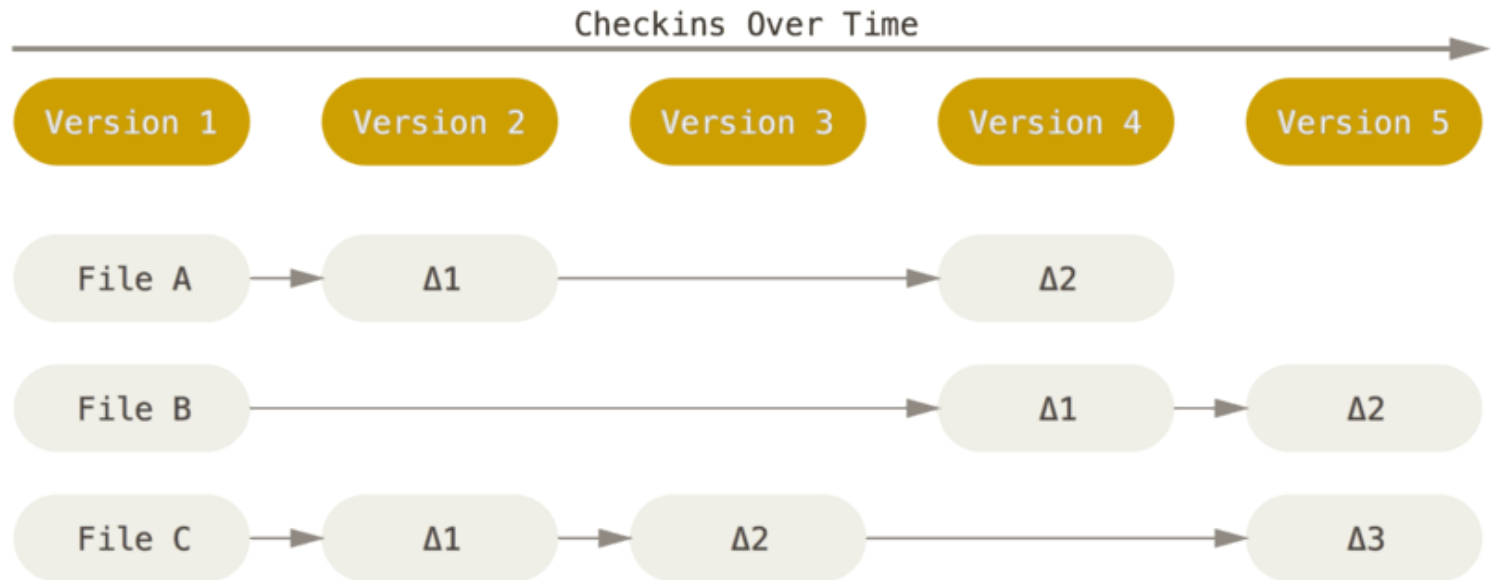
-b, --branch

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

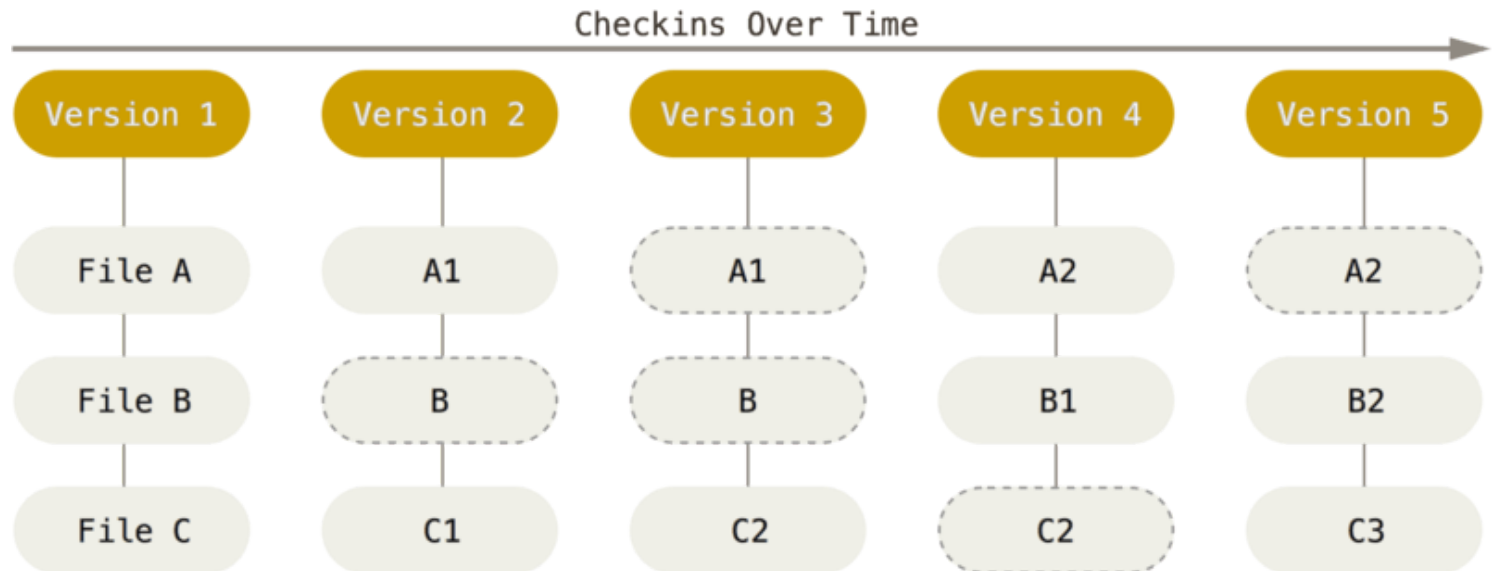
Main ideas in Git

- 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

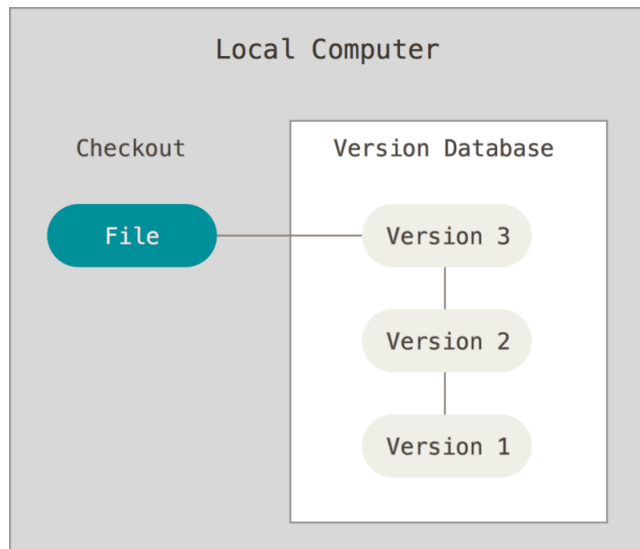


- Snapshots (Git)



2. Git basic usage

Using git as a local VCS



Create a repository

git init

In [6]:

```
%%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 Nov 12 13:45 .
```

```
drwxr-xr-x 7 angelv angelv 4096 Nov 12 13:45 ..
```

```
drwxr-xr-x 7 angelv angelv 4096 Nov 12 13:45 .git
```

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

In [7]:

```
%%bash
```

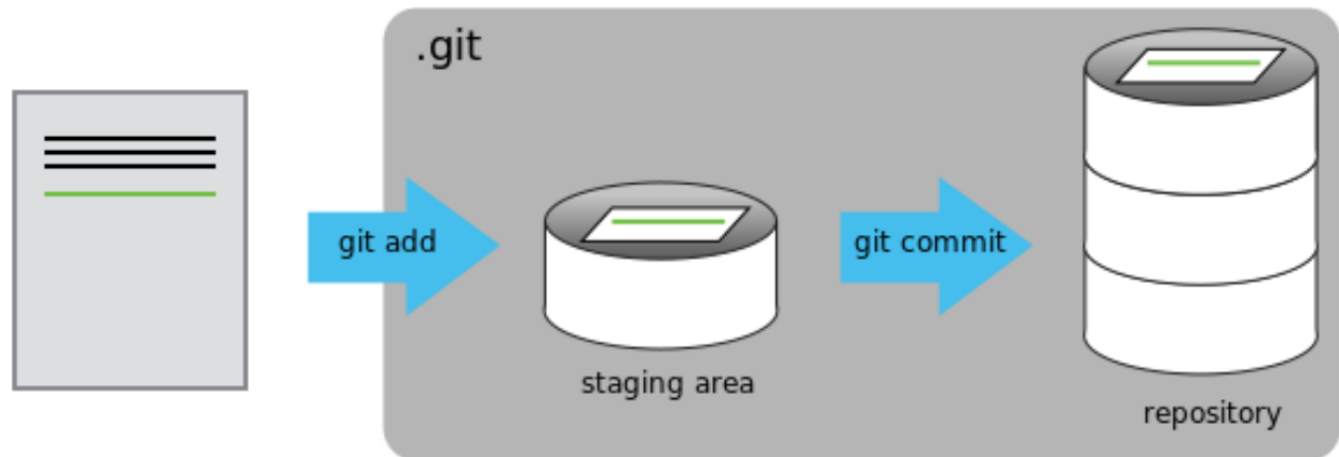
```
cd ../Demos/Repos  
ls First_Demo/.git
```

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

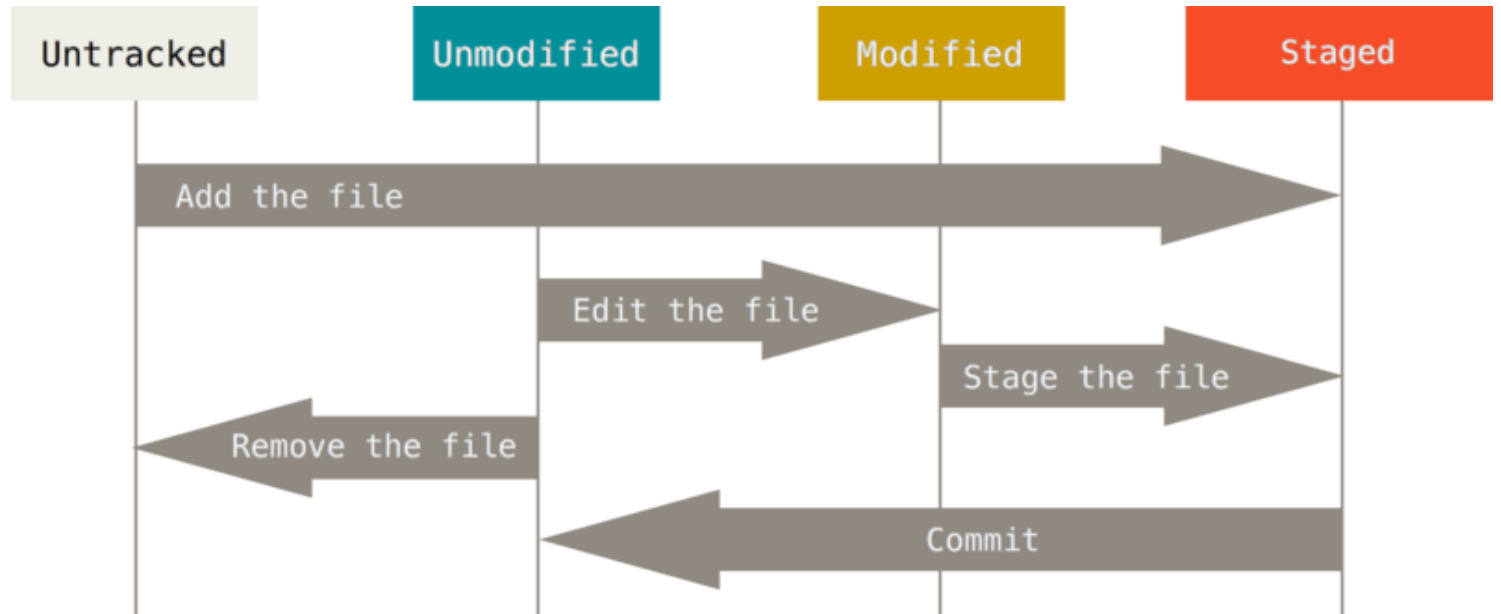
Recording changes

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

- Committing changes



- Tracking changes



Recording changes demo

[https://asciinema.org/a/AWQng1KOIEOxsCe1qdpadv0sN?
autoplay=1&cols=180&rows=40](https://asciinema.org/a/AWQng1KOIEOxsCe1qdpadv0sN?autoplay=1&cols=180&rows=40)
([https://asciinema.org/a/AWQng1KOIEOxsCe1qdpadv0sN?
autoplay=1&cols=180&rows=40](https://asciinema.org/a/AWQng1KOIEOxsCe1qdpadv0sN?autoplay=1&cols=180&rows=40)).

Viewing history

- `git log`
- `git diff`

Viewing history demo

[https://asciinema.org/a/Kw55IEVIHUFOGauWs8vW0RQ04?
autoplay=1&cols=180&rows=40](https://asciinema.org/a/Kw55IEVIHUFOGauWs8vW0RQ04?autoplay=1&cols=180&rows=40)
([https://asciinema.org/a/Kw55IEVIHUFOGauWs8vW0RQ04?
autoplay=1&cols=180&rows=40](https://asciinema.org/a/Kw55IEVIHUFOGauWs8vW0RQ04?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

[https://asciinema.org/a/IlbHbix088FqwazA2Ms7hJZT6?
autoplay=1&cols=180&rows=40](https://asciinema.org/a/IlbHbix088FqwazA2Ms7hJZT6?autoplay=1&cols=180&rows=40)
([https://asciinema.org/a/IlbHbix088FqwazA2Ms7hJZT6?
autoplay=1&cols=180&rows=40](https://asciinema.org/a/IlbHbix088FqwazA2Ms7hJZT6?autoplay=1&cols=180&rows=40))

- In older versions of `git`, instead of `git restore` you would use:

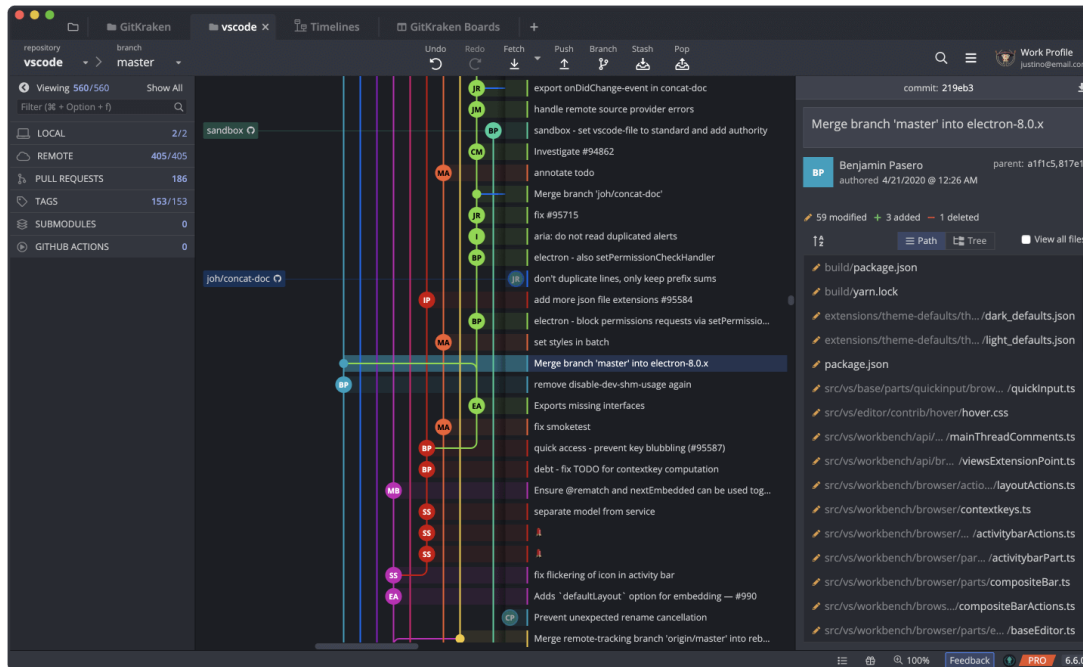
- `git reset HEAD ...`
- and `git checkout ...`

(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>))

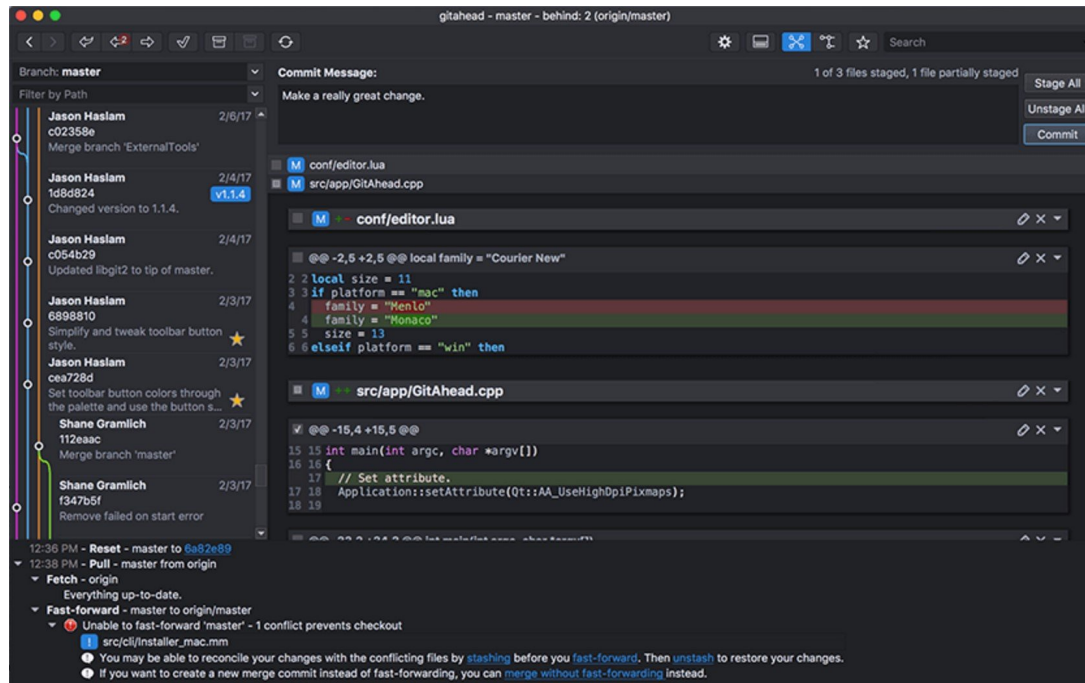
Command-line vs GUIs

- 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/>).

	Head:	master Merge pull request #257 from jotoeri/add-gullink
3888d9c	Merge:	origin/master Merge pull request #257 from jotoeri/add-gullink
7934d99	Tag:	4.4.0 (17)
	Recent commits	
aaf748d	master origin/master Merge pull request #257 from jotoeri/add-gullink	
7934d99	Adding a link to the GUI overview	jotoeri 1 year
aaf748d	Merge pull request #252 from 0x15f9/master	Jessica Lord 1 year
7b389ca	* Fixed slight typo	0x15f9 1 year
8c648b2	Merge pull request #242 from franciosi/patch-2	Jessica Lord 1 year
9eba724	Update README.md	Franciosi 1 year
aac8592	Merge pull request #238 from juanuarui/patch-1	Jessica Lord 2 years
07d59ee	Cambiando "and" por "y"	Juan Antonio Ruiz... 2 years
8dc8753	* Accurate - Asignate	Jessica Lord 2 years
8a2594f	Merge pull request #234 from diegocasilas/spanish-translati	Jessica Lord 2 years
0d35369	* Fix typos in 11_merge_tada.html	diegocasilas 2 years
3de27cb	* Fix typos in 10_requesting you, pull please.html	diegocasilas 2 years
183256b	* Fix typos in 9_pull_never_out_of_date.html	diegocasilas 2 years
90e0692	* Fix typos in 7_branches_arent_just_for_birds.html	diegocasilas 2 years
080b955	* Fix typos in 6_forks_and_clones.html	diegocasilas 2 years
aac8481	* Fix typos in 5_remote_control.html	diegocasilas 2 years
0d4cbe	* Fix typos in 4_githubbin.html	diegocasilas 2 years
6bc3829	4.4.0 Merge pull request #231 from mberasategi/es ES	Jessica Lord 2 years
elc5963	+ Add es-ES to README	mberasategi 2 years
05155c6	+ Set es-ES for es alias	mberasategi 2 years
18316a1	+ Add new locale to menus	mberasategi 2 years
86766da	- Translate pages	mberasategi 2 years
2a1c1a1	- Translate buttons	mberasategi 2 years
86bd244	- Translate 11_merge	mberasategi 2 years
1b78e0d	- Translate 10-requesting-pullrequest	mberasategi 2 years
084568c	- Translate 9-pull	mberasategi 2 years
02c789c	- Translate 8-small-world	mberasategi 2 years
085331b	- Translate 7-branches	mberasategi 2 years
e559315	- Translate 6-forks-clones	mberasategi 2 years
1a45398	- Translate 5-remote-control	mberasategi 2 years
0cd4dbae	- Translate 4-gitbin	mberasategi 2 years
370610d	- Translate 3-commit-to-it	mberasategi 2 years
06d226f	- Change hello-world to hola-mundo	mberasategi 2 years
0911c4e	- Change Git Shell to git_PWD for Windows	mberasategi 2 years
0172317	- Begin translating 3-commit-to-it	Niren Berasategi 2 years
016927f	- Translate 2-repository	Niren Berasategi 2 years
0eb221b	- Translate 1-get	Niren Berasategi 2 years
1c456dd	- Duplicate es_CO to jumpstart es_ES	Niren Berasategi 2 years
7aa55c5	+ Add list of supported languages to readme	Jessica Lord 2 years
2730989	+ 4.3.0 version 4.3.0	Jessica Lord 2 years
81f9870	+ make lang button wider to fit language names	Jessica Lord 2 years
71c691	Merge pull request #217 from node-co/feature/spanish-translati	Jessica Lord 2 years
0910665F21	magit-log: git-ref: Top (5,9) Magit Log Ivy Projectile yas	86324.0.61 -----
	99-UUW:0A%-F21 magit: git-if-electron All (8,0) (Magit Ivy Projectile yas)	8634.0.61 -----
	Author: Jessica Lord <jlord@glitch.com>	
	AuthDate: Sun Mar 10 20:49:34 2019 -0400	
	Commit: GITHub -enop IvygitHub.com	
	ComDate: Sun Mar 10 20:49:34 2019 -0400	
	Parent: 8c648b2 Merge pull request #242 from franciosi/patch-2	
	Parent: f3389ca 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/challenges4_githubbin.html 4 +---	
	back	
	99-UUW:0A%-F21 magit-revision: git-if-electron All (3,0) (Magit Rev Ivy Projectile yas)	8634.0.61 -----

3. Branches

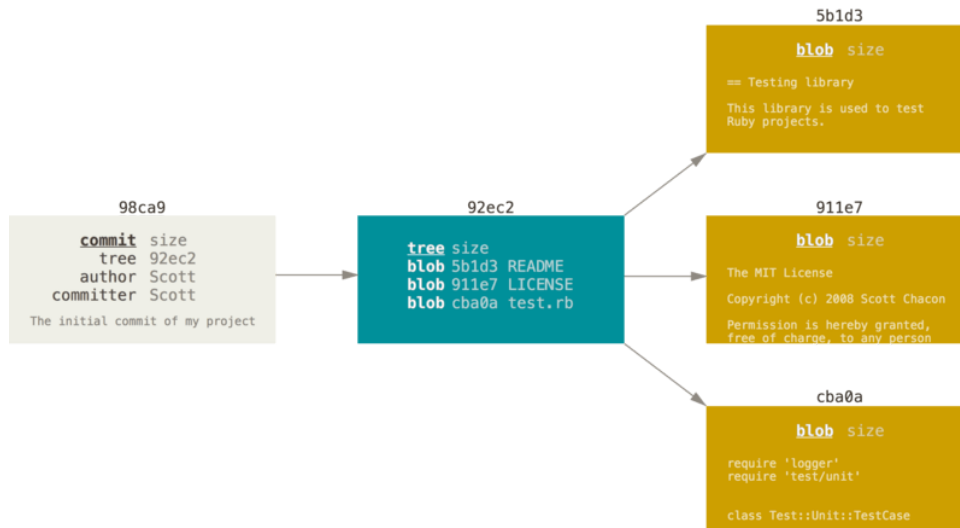
Branches

- 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.

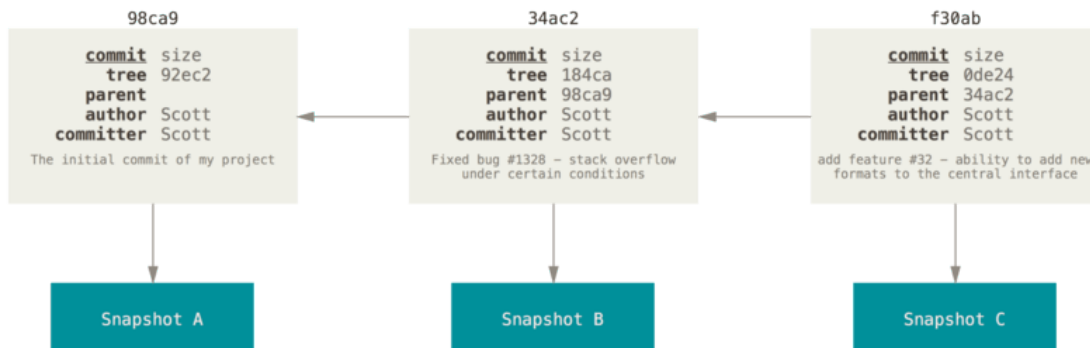
So, let's try and see some of the Git internals to understand what branches are...

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

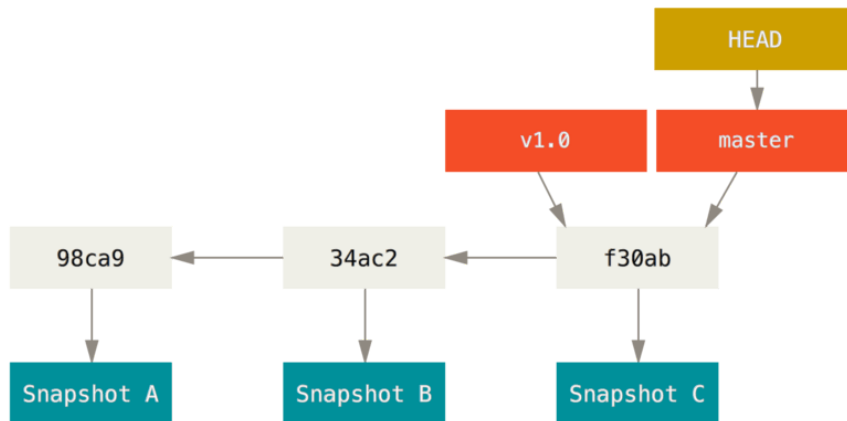
- A *commit* points to a *tree* object, and this to *blobs*
 - For us the important part is just the *commit*, which points to a *snapshot* of our work



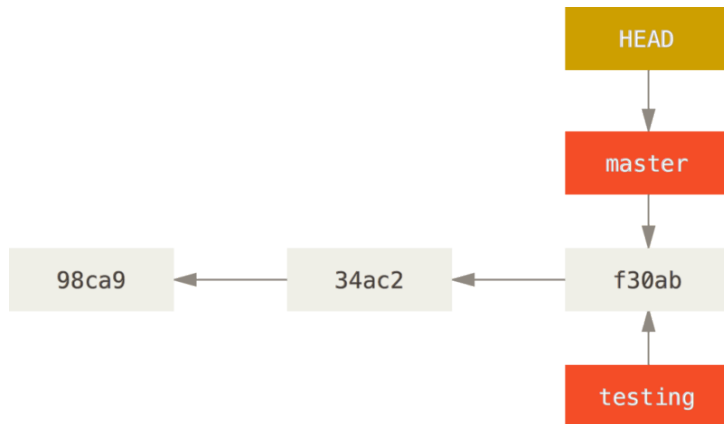
- Creating more *commits*, keeps a linked list
 - (which is basically what you see when you do `git log`)



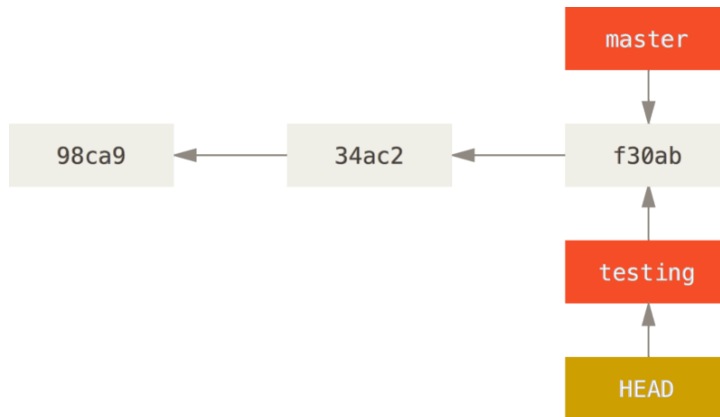
- A *branch* is just a pointer to one of these *commits*
 - *master* is created when you do `git init`, but it is no special
 - *HEAD* points to the branch we are working on



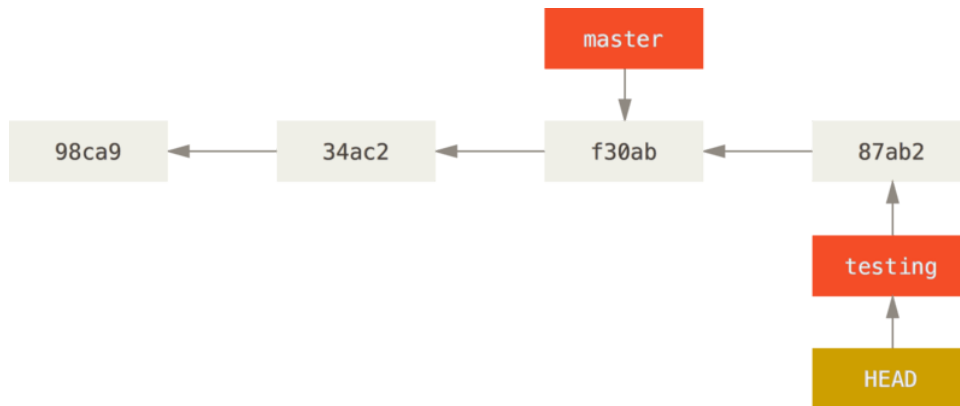
- Creating a branch
 - If we are working in *master* and we do `git branch testing`
 - *testing* branch is created, but we are still working on *master*
 - so you can see *HEAD* is still pointing to *master*



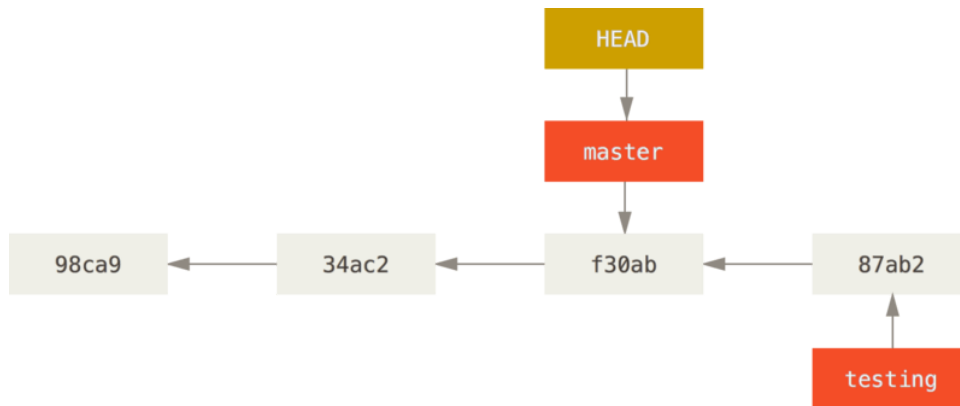
- To work with another branch
 - `git checkout testing` makes *HEAD* point to testing
 - when you switch branches in Git:
 - **files in your working directory will change.**
 - **if Git cannot do it cleanly, it will not let you switch at all.**



- 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)



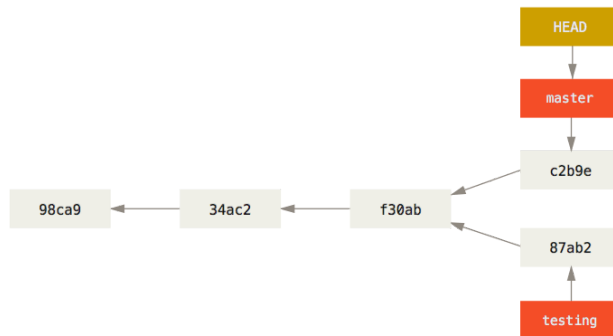
Branching demo

- 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*)

<https://asciinema.org/a/C8RYb5n0IFitsEw94c2dDifZU?autoplay=1&cols=180&rows=40>
([https://asciinema.org/a/C8RYb5n0IFitsEw94c2dDifZU?](https://asciinema.org/a/C8RYb5n0IFitsEw94c2dDifZU?autoplay=1&cols=180&rows=40)
[autoplay=1&cols=180&rows=40](https://asciinema.org/a/C8RYb5n0IFitsEw94c2dDifZU?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*)

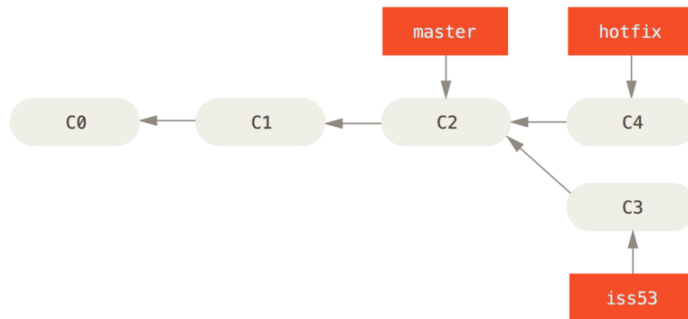
4. Merging

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

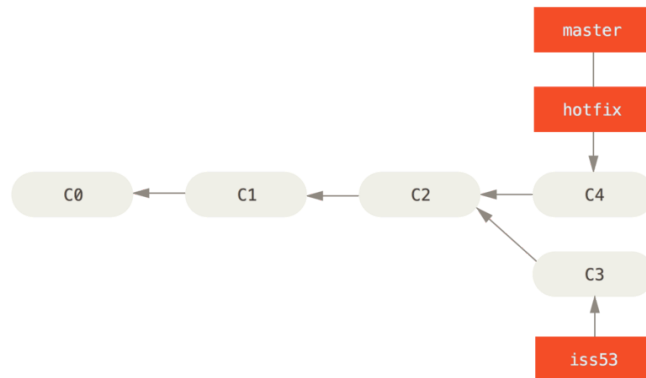
Fast-forward "merge"

- 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)

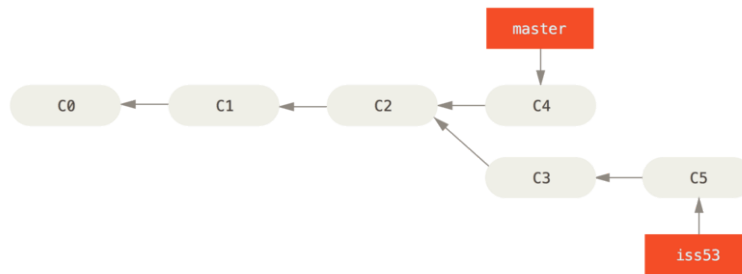
- Check out the master branch and merge the hotfix branch
 - This is a fast-forward merge (i.e. nothing really to merge, not divergent branch)



<https://asciinema.org/a/nhtnwR2msl8sE4TEf2WwryQSU?autoplay=1&cols=180&rows=40>
(<https://asciinema.org/a/nhtnwR2msl8sE4TEf2WwryQSU?autoplay=1&cols=180&rows=40>)

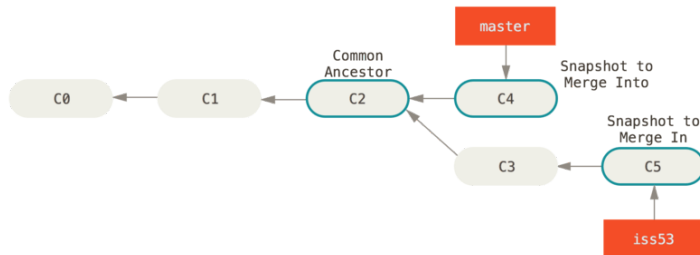
No-conflicts merge

- 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*



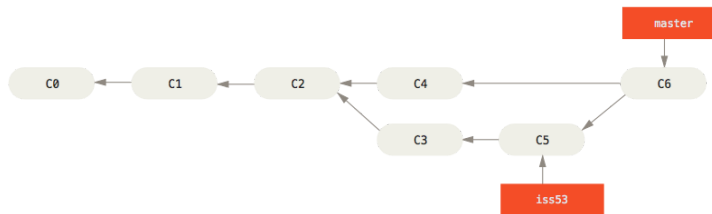
No-conflicts merge (2)

- 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)

[https://asciinema.org/a/9oGkREQonLUPQTkfxZQebdipL?
autoplay=1&cols=180&rows=40](https://asciinema.org/a/9oGkREQonLUPQTkfxZQebdipL?autoplay=1&cols=180&rows=40)
([https://asciinema.org/a/9oGkREQonLUPQTkfxZQebdipL?
autoplay=1&cols=180&rows=40](https://asciinema.org/a/9oGkREQonLUPQTkfxZQebdipL?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

<https://asciinema.org/a/kc1x2LpQM0e2g1Jw2EDWGFTHm?autoplay=1&cols=180&rows=40>
[\(https://asciinema.org/a/kc1x2LpQM0e2g1Jw2EDWGFTHm?autoplay=1&cols=180&rows=40\)](https://asciinema.org/a/kc1x2LpQM0e2g1Jw2EDWGFTHm?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>
[\(https://www.youtube.com/watch?v=W-FHwUwE84M\)](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)

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>).