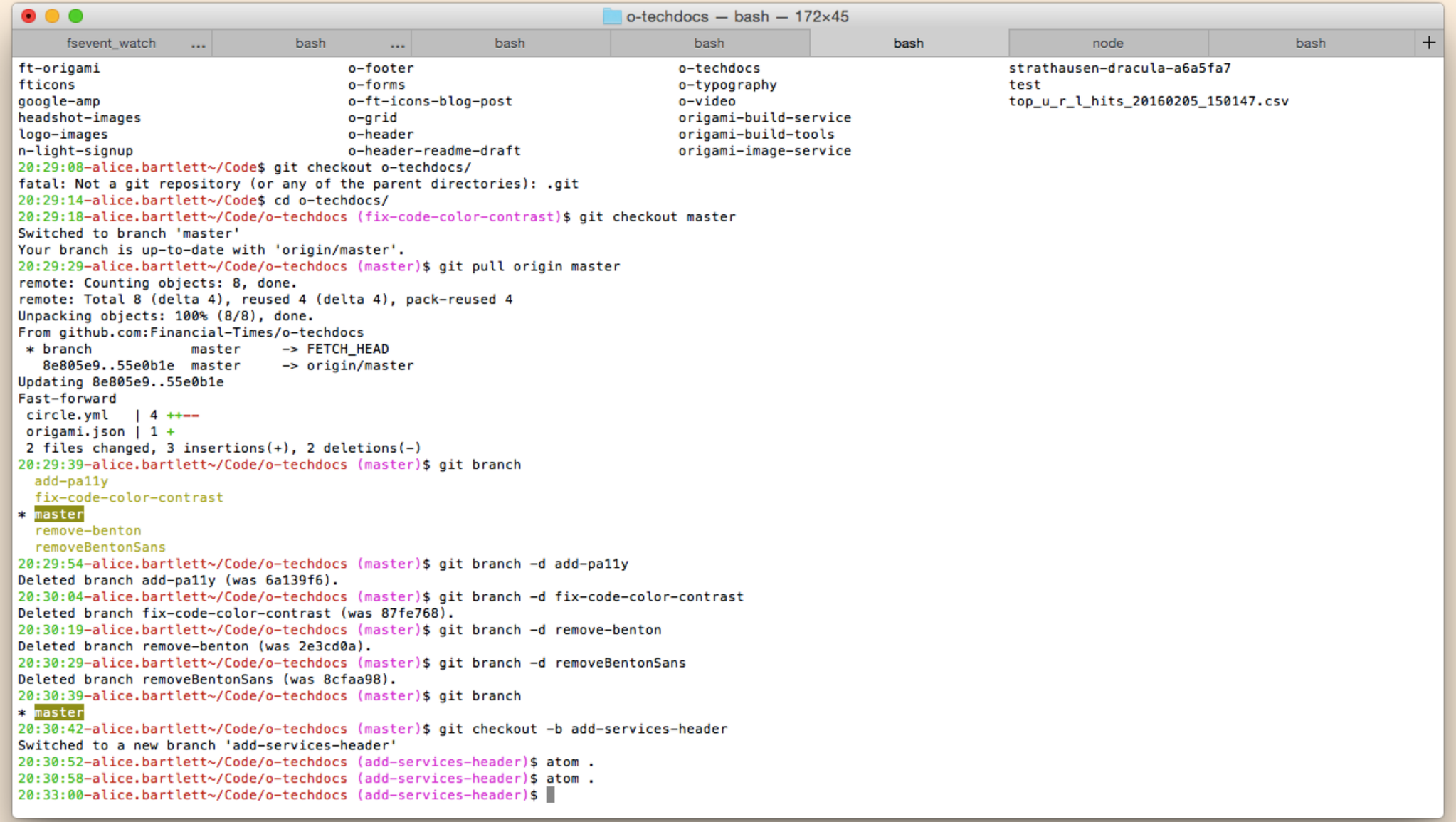
WHAT IS GIT

WHAT DOES IT DO?

Git helps you manage work done on projects.

GIT IS UNFRIENDLY


o-techdocs — bash — 172×45
fsevent_watch
bash
bash
bash
bash
node
bash
ft-origami
fticons
google-amp
headshot-images
logo-images
n-light-signup
o-footer
o-forms
o-ft-icons-blog-post
o-grid
o-header
o-header-readme-draft
o-techdocs
o-typography
o-video
origami-build-service
origami-build-tools
origami-image-service
strathausen-dracula-a6a5fa7
test
top_u_r_l_hits_20160205_150147.csv
20:29:08-alice.bartlett~/Code$ git checkout o-techdocs/
fatal: Not a git repository (or any of the parent directories): .git
20:29:14-alice.bartlett~/Code$ cd o-techdocs/
20:29:18-alice.bartlett~/Code/o-techdocs (fix-code-color-contrast)$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
20:29:29-alice.bartlett~/Code/o-techdocs (master)$ git pull origin master
remote: Counting objects: 8, done.
remote: Total 8 (delta 4), reused 4 (delta 4), pack-reused 4
Unpacking objects: 100% (8/8), done.
From github.com:Financial-Times/o-techdocs
 * branch   master   -> FETCH_HEAD
   8e805e9..55e0b1e master   -> origin/master
Updating 8e805e9..55e0b1e
Fast-forward
 circle.yml | 4 ++--
 origami.json | 1 +
 2 files changed, 3 insertions(+), 2 deletions(-)
20:29:39-alice.bartlett~/Code/o-techdocs (master)$ git branch
  add-pa11y
  fix-code-color-contrast
* master
  remove-benton
  removeBentonSans
20:29:54-alice.bartlett~/Code/o-techdocs (master)$ git branch -d add-pa11y
Deleted branch add-pa11y (was 6a139f6).
20:30:04-alice.bartlett~/Code/o-techdocs (master)$ git branch -d fix-code-color-contrast
Deleted branch fix-code-color-contrast (was 87fe768).
20:30:19-alice.bartlett~/Code/o-techdocs (master)$ git branch -d remove-benton
Deleted branch remove-benton (was 2e3cd0a).
20:30:29-alice.bartlett~/Code/o-techdocs (master)$ git branch -d removeBentonSans
Deleted branch removeBentonSans (was 8cfaa98).
20:30:39-alice.bartlett~/Code/o-techdocs (master)$ git branch
* master
20:30:42-alice.bartlett~/Code/o-techdocs (master)$ git checkout -b add-services-header
Switched to a new branch 'add-services-header'
20:30:52-alice.bartlett~/Code/o-techdocs (add-services-header)$ atom .
20:30:58-alice.bartlett~/Code/o-techdocs (add-services-header)$ atom .
20:33:00-alice.bartlett~/Code/o-techdocs (add-services-header)$

GIT

UNDERNEATH ALL THIS, GIT IS QUITE SIMPLE

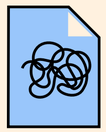
1. THING 1
2. THING 2
3. THING 3
4. THING 4
5. THING 5

THING 1:

GIT LETS YOU TELL THE STORY OF YOUR PROJECT

You use Git to take snapshots of all the files in a folder.
This folder is called a repository or repo.

When you want to take a snapshot of a file or files, you create a commit



logo.svg

logo-2.svg

logo-3-monica-feedback.svg

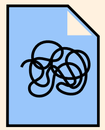
logo-3-FINAL.svg

logo-3-FINAL-1.svg

By saving copies

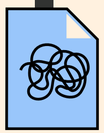
By making commits

By saving copies



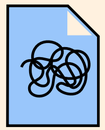
`logo.svg`

By making commits



`logo.svg`

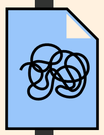
By saving copies



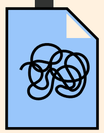
`logo.svg`

`logo-2.svg`

By making commits

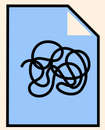


`logo.svg`



`logo.svg`

By saving copies

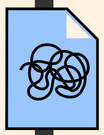


`logo.svg`

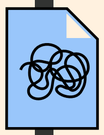
`logo-2.svg`

`logo-3-monica-feedback.svg`

By making commits



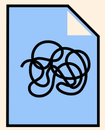
`logo.svg`



`logo.svg`

`logo.svg`

By saving copies



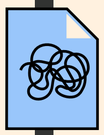
`logo.svg`

`logo-2.svg`

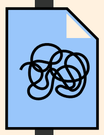
`logo-3-monica-feedback.svg`

`logo-3-FINAL.svg`

By making commits

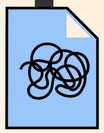


`logo.svg`



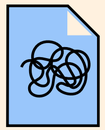
`logo.svg`

`logo.svg`



`logo.svg`

By saving copies

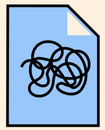


`logo.svg`

`logo-2.svg`

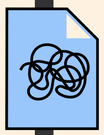
`logo-3-monica-feedback.svg`

`logo-3-FINAL.svg`

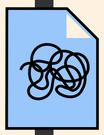


`logo-3-FINAL-1.svg`

By making commits

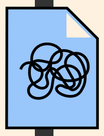


`logo.svg`

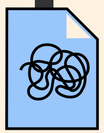


`logo.svg`

`logo.svg`

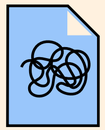


`logo.svg`



`logo.svg`

By saving copies

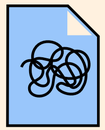


`logo.svg`

`logo-2.svg`

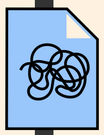
`logo-3-monica-feedback.svg`

`logo-3-FINAL.svg`

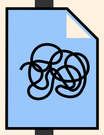


`logo-3-FINAL-1.svg`

By making commits

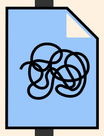


`logo.svg`

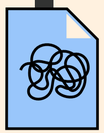


`logo.svg`

`logo.svg`



`logo.svg`



`logo.svg`

When you commit a file or files, some information is saved along with the changes to the file

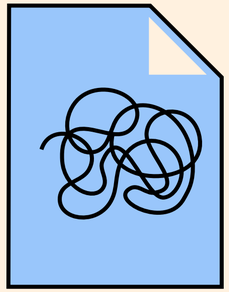
1. Who
2. When

You can add more information about the changes you've made in a commit message

A good commit message:

```
Update link style

User research showed that many people did not
spot links in the copy. This commit updates the
link style to the new underlined style which
performed better.
```



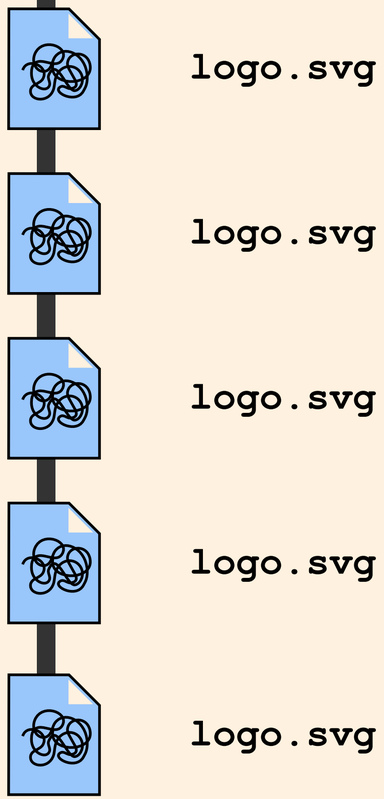
`logo-3-FINAL-1.svg`

Alice Bartlett
10:34am March 11th 2016

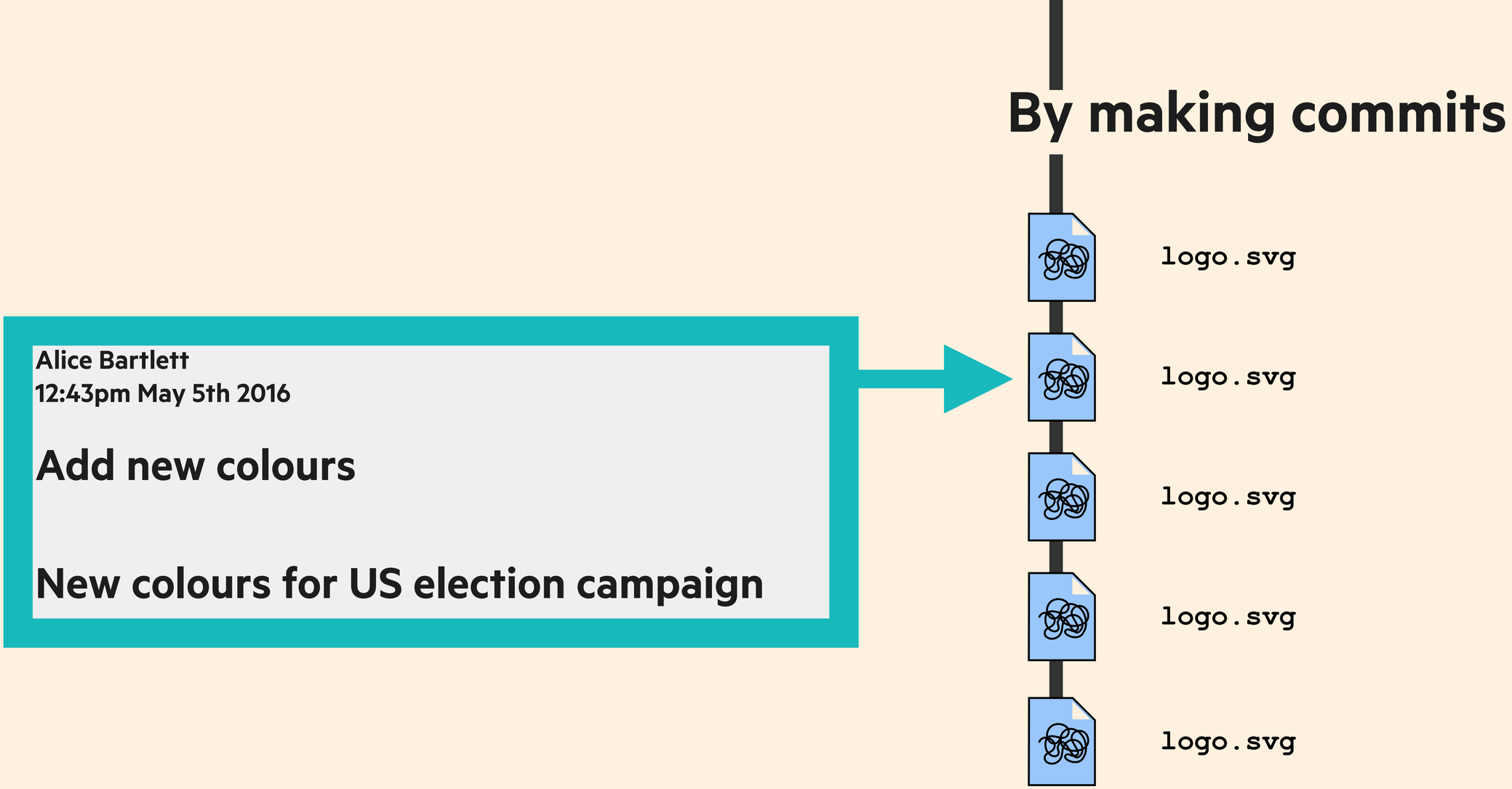
Update link style

User research showed that many people did not spot links in the copy. This commit updates the link style to the new underlined style which performed better.

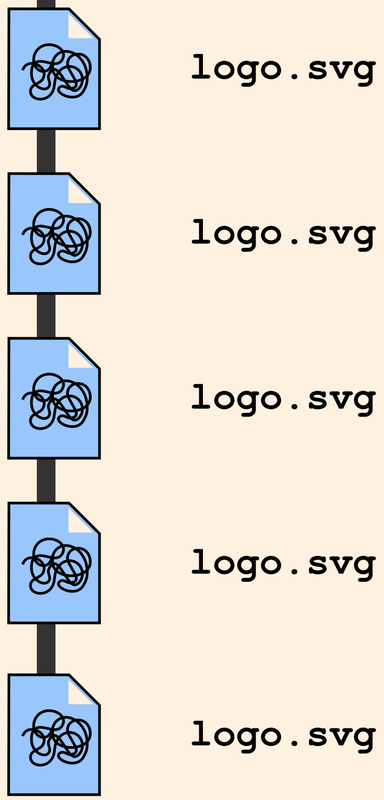
By making commits



By making commits



By making commits

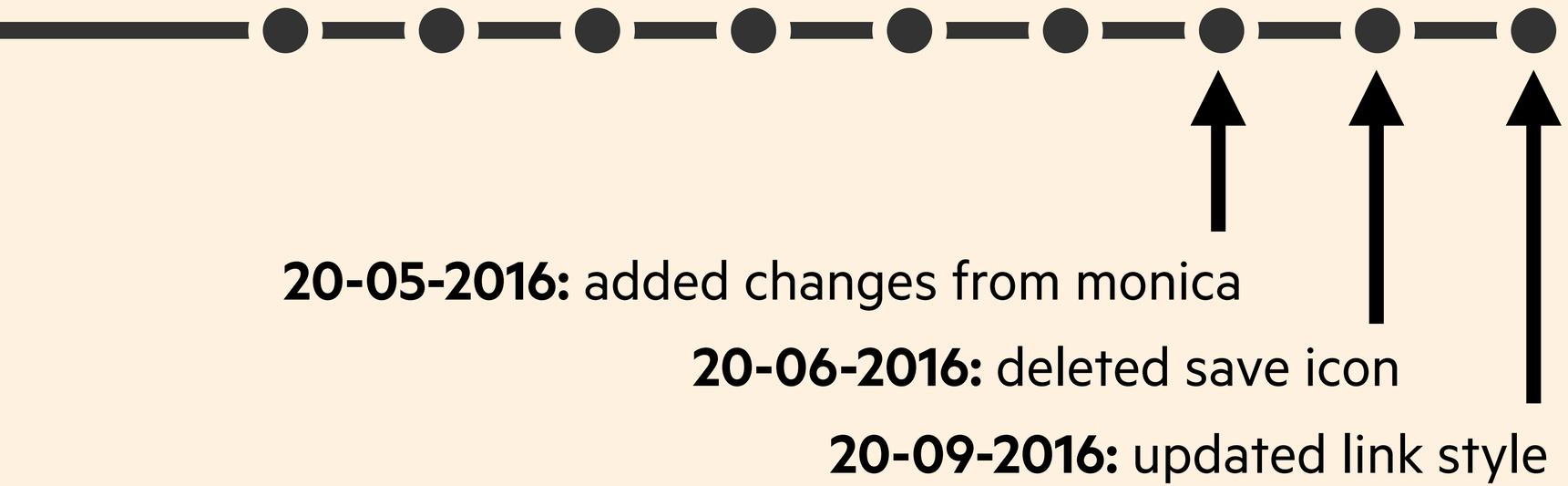


Alice Bartlett
12:43pm May 8th 2016

Fix Orange

The orange we used fails AAA accessibility contrast tests so beef it up to contrast properly

Git stores the whole history of your project



repository - your project folder

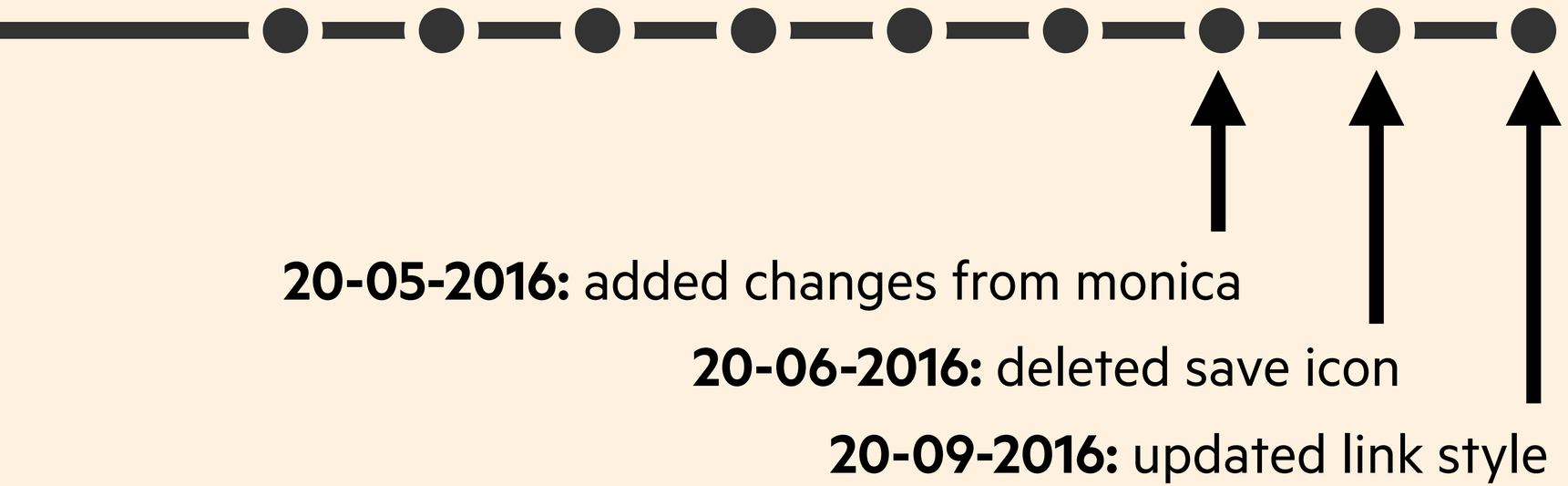
commit - save a snapshot

THING 2:

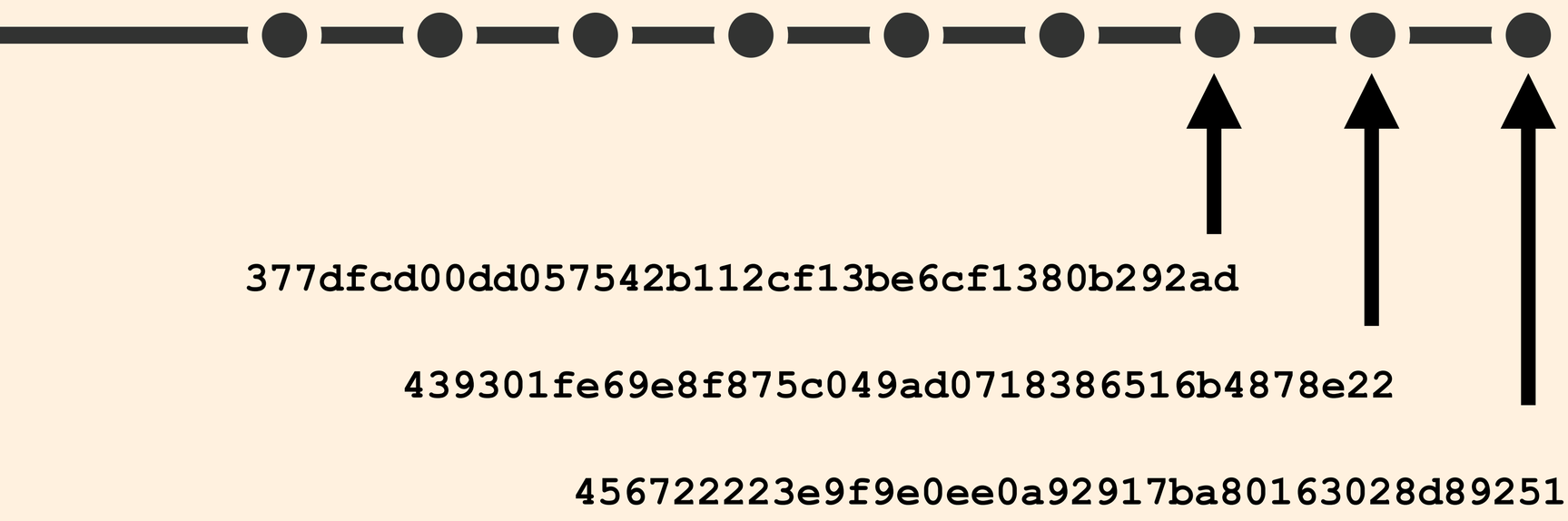
GIT LETS YOU TIME TRAVEL

Once you’ve saved some snapshots, Git lets you move through them

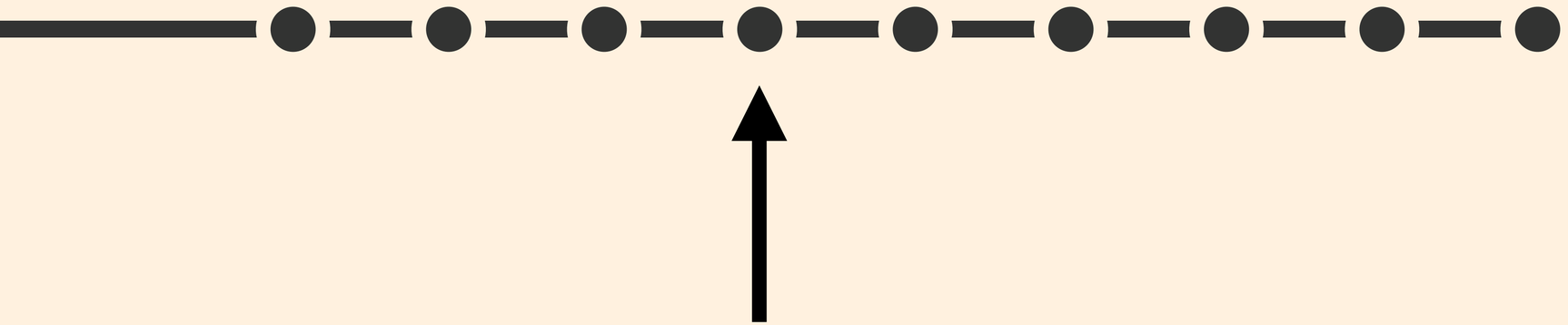
Git stores the whole history of your project



Each of these commits has an id called a hash



I can tell Git what commit I want to check out using the commit hash

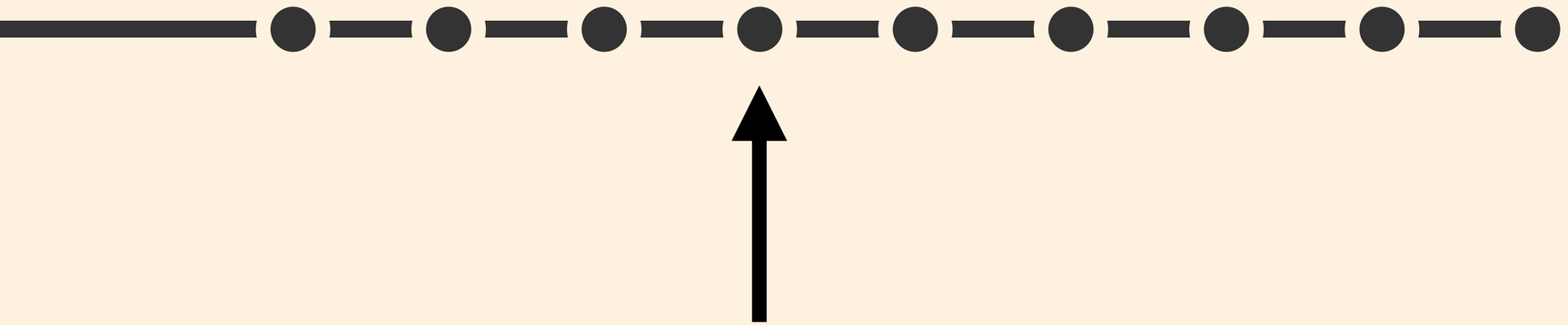


20-05-2016: deleted play icon

`d5b87865bc2cd9d38ba8284c2eaa0d0241d800bb`

Getting the files from a commit in the past is known as doing a check out

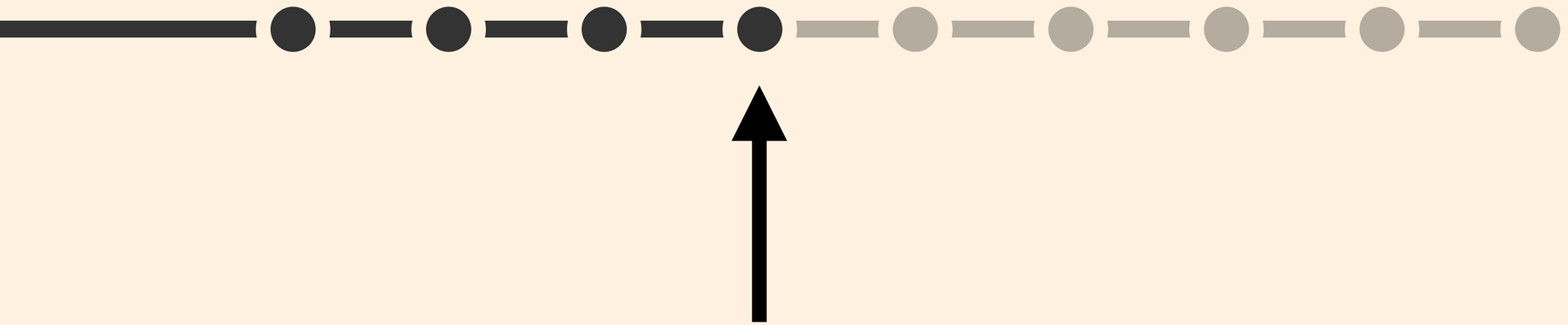
I can tell Git what commit I want to check out using the commit hash



20-05-2016: deleted play icon

`d5b87865bc2cd9d38ba8284c2eaa0d0241d800bb`

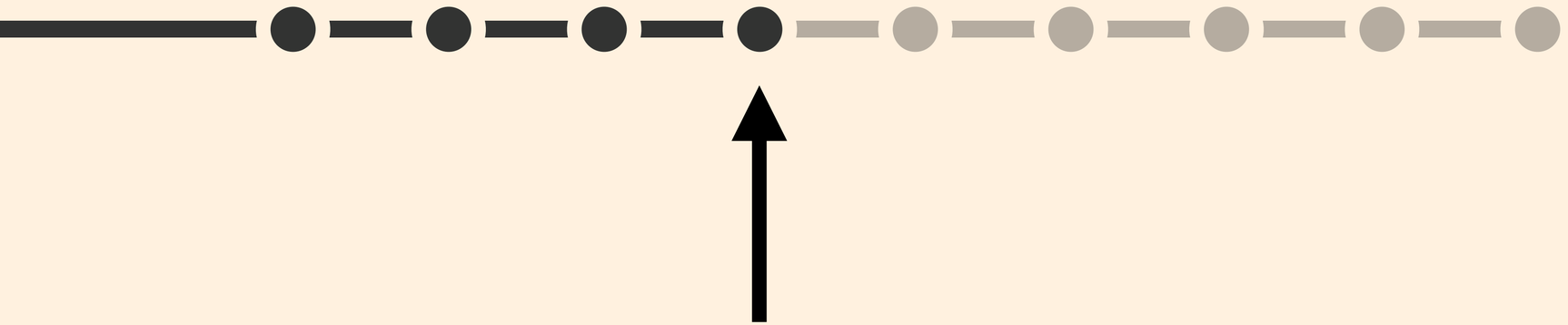
I can tell Git what commit I want to check out using the commit hash



20-05-2016: deleted play icon

`d5b87865bc2cd9d38ba8284c2eaa0d0241d800bb`

My other commits still exist, but when I look in my repo, it's as if they never happened



20-05-2016: deleted play icon

`d5b87865bc2cd9d38ba8284c2eaa0d0241d800bb`

hash - a computer generated id

checkout - time travel to a specific commit

THING 3:

GIT HELPS YOU EXPERIMENT

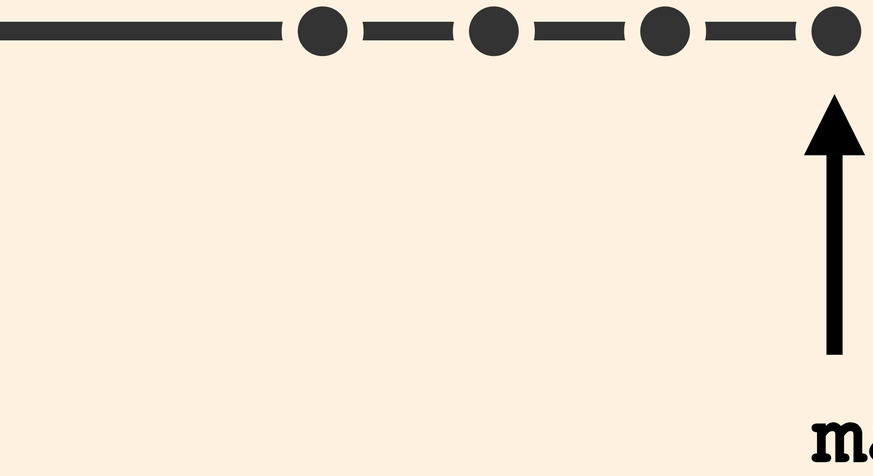
So far, everything has been very linear and ordered.

This isn’t really how projects work, sometimes you want to make easily discardable experiments

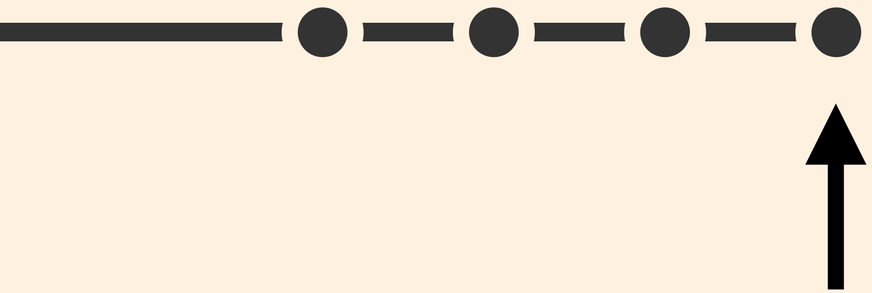
The way you do this in Git is with branches

A branch is a moveable label attached to a commit

The default branch name in Git is master



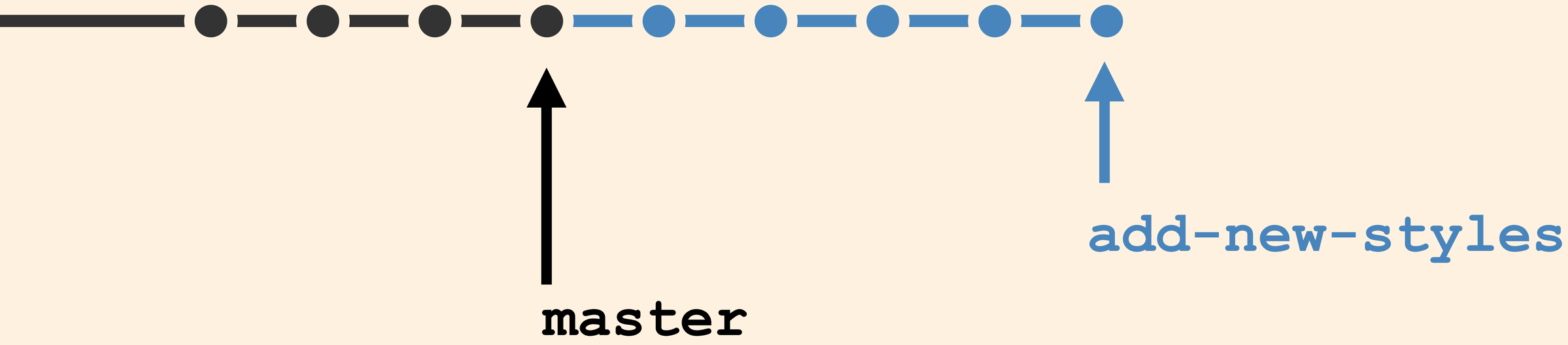
You can add your own branches too

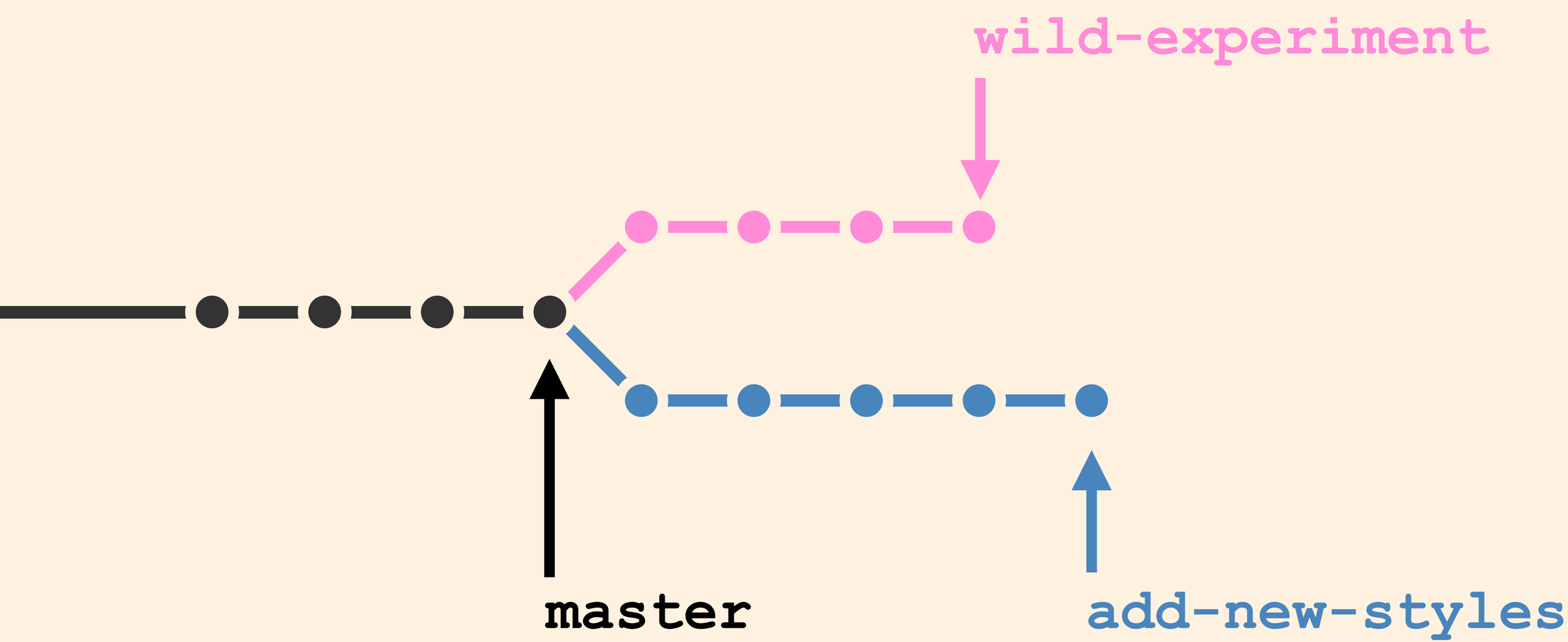


`add-new-styles`

`master`

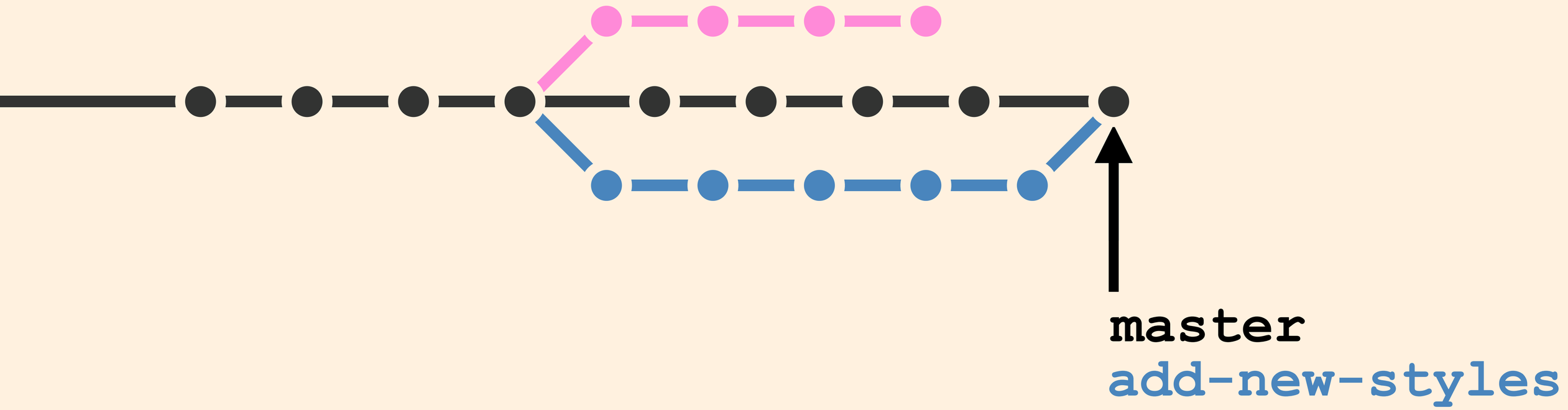
A developer will often do lots of work on a branch



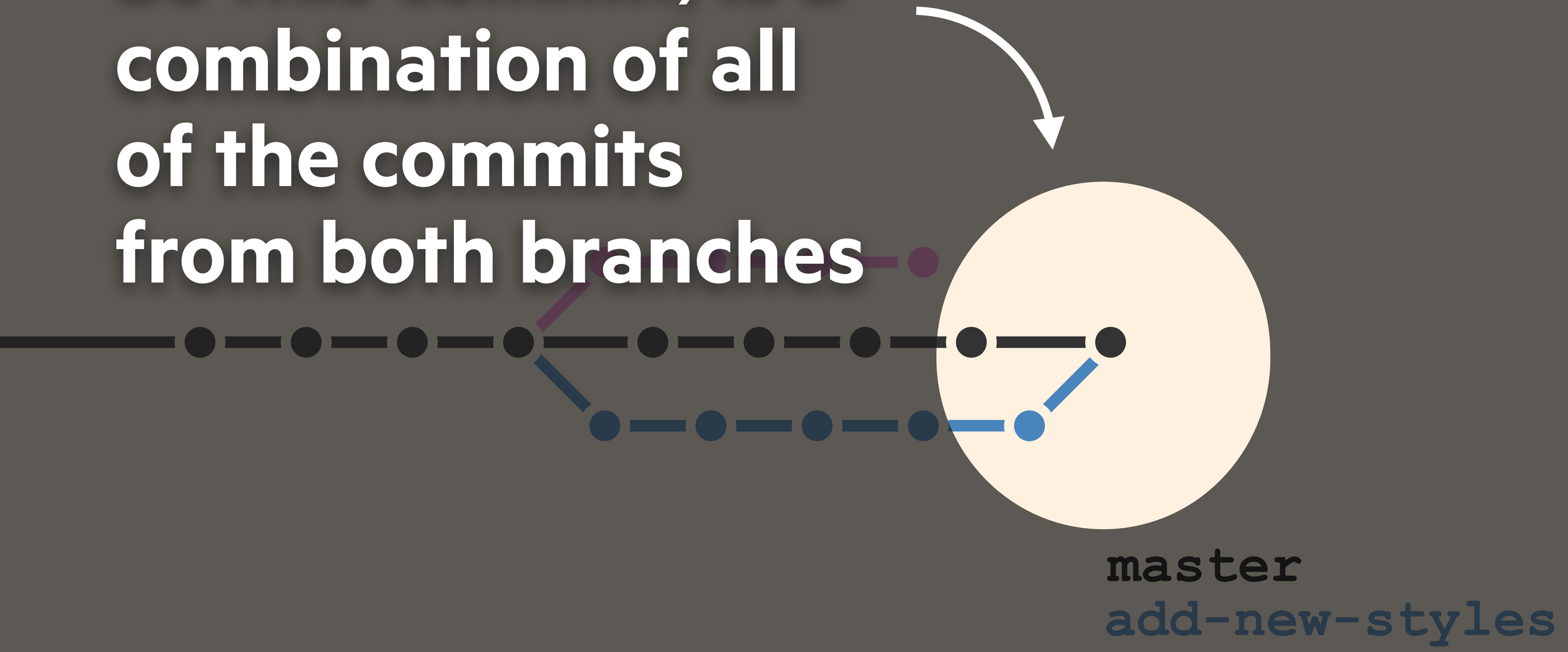

wild-experiment
master
add-new-styles

Once you’re happy with some work, you need a way to get it back into master

To get changes from one branch into another, you merge them



So this commit, is a combination of all of the commits from both branches



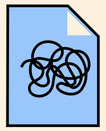
branch - a moveable label that points to a commit
merge - the combination of two or more branches

THING 4:

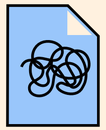
GIT HELPS YOU BACK UP YOUR WORK

Everyone knows that you should back up your work regularly

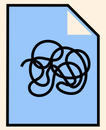
Ideally to somewhere that is geographically distinct from your computer



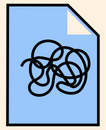
`logo.svg`



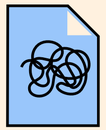
`logo-2.svg`



`logo-3-monica-feedback.svg`



`logo-3-FINAL.svg`



`logo-3-FINAL-1.svg`

- Safer
- Access from different places
- Shared access

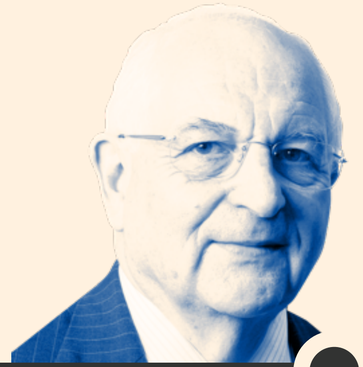
In Git this place is called a remote

A very popular remote is Github

To get some work from a remote for the first time you clone it

Remote

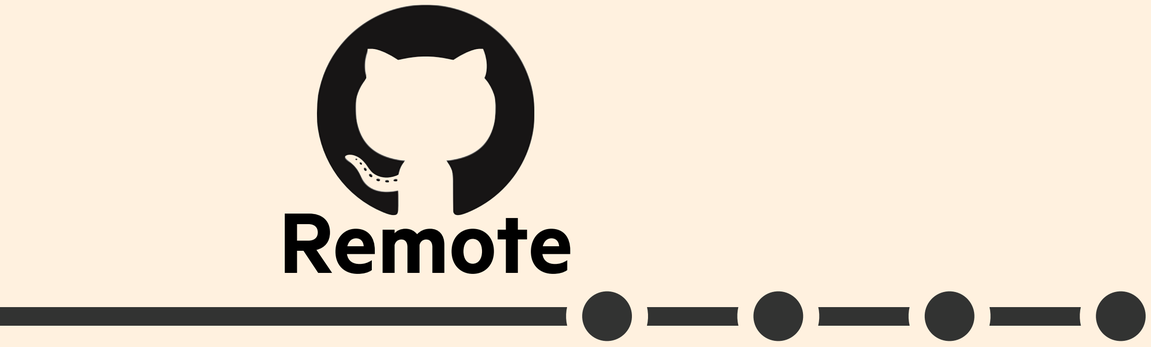
Remote

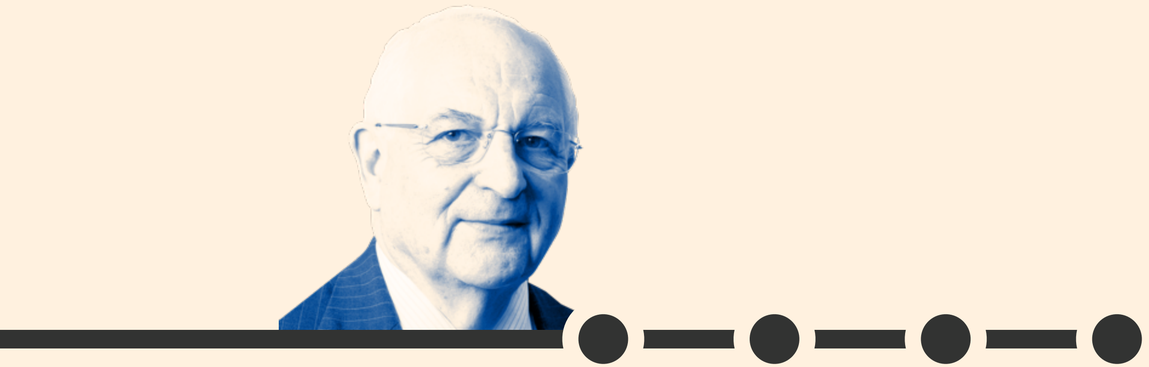


clone

Remote

clone





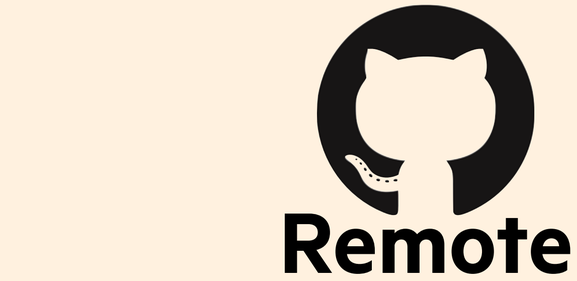
Now everyone has the repo on their computer

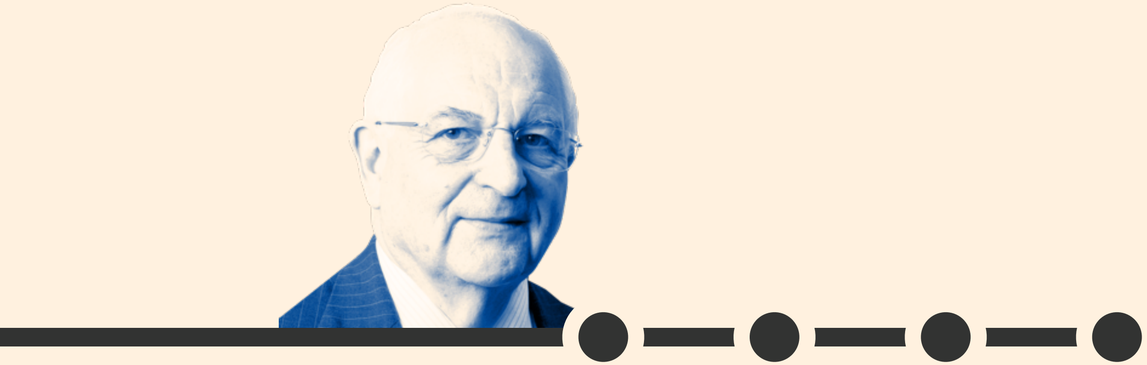
Remote

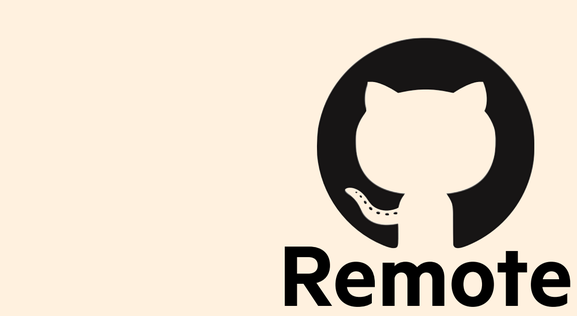
Lucy Kellaway
10:34am November 4th 2016

Fix broken icon tinting

Icon tinting was case sensitive so #FFF worked but #fff didn't. This commit removes this bug.


Remote

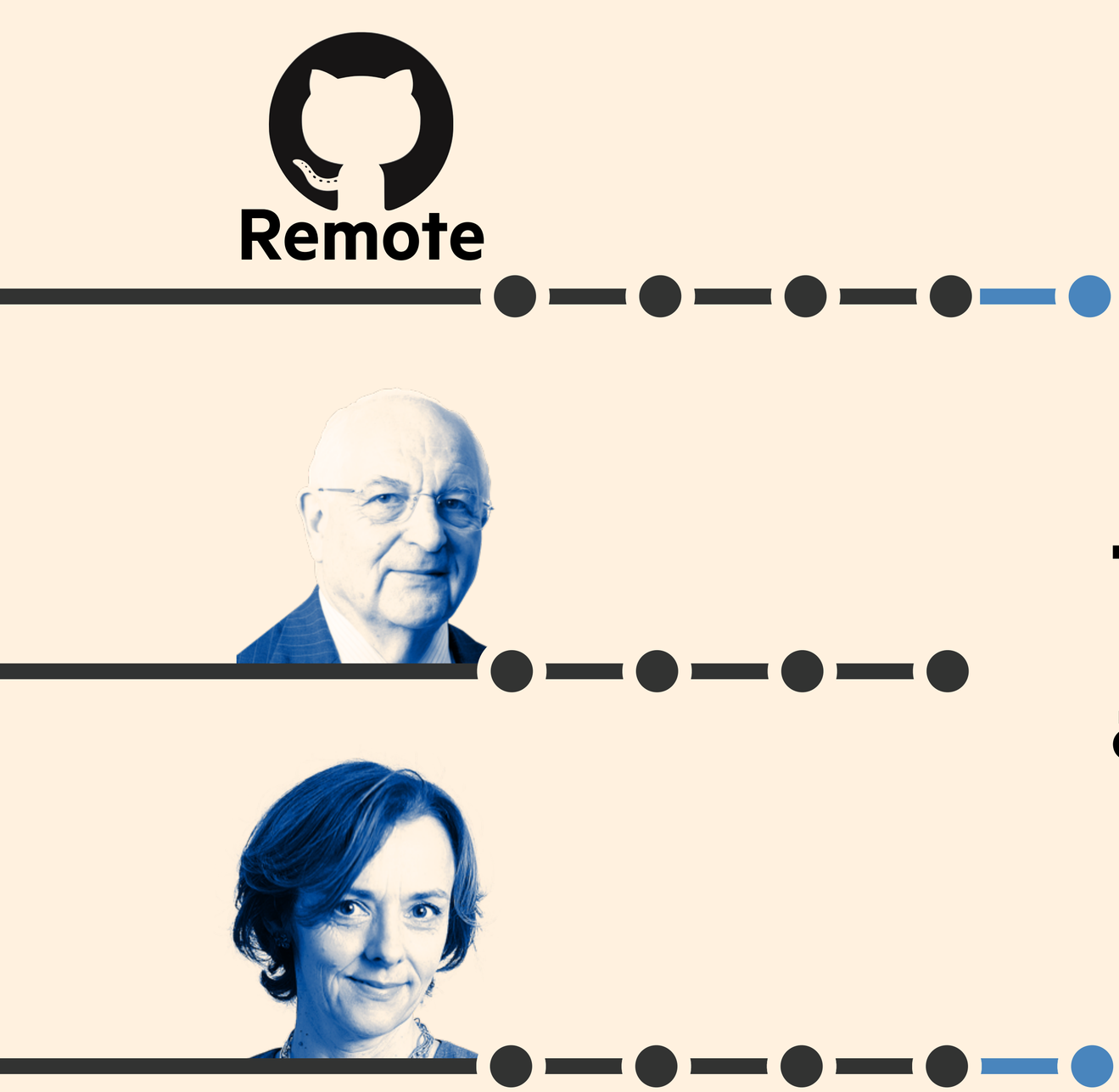




Lucy can send her changes to remote

Remote

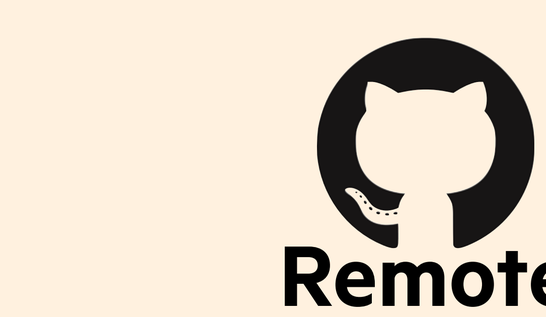
push!



This is known as a push

Remote

Now Martin is behind



To get these changes, Martin will need to pull them

Remote

pull!

remote - a computer with a repo on it
clone - get the repo from the remote for the first time
pull - get new commits to the repo from the remote
push - send your new commits to the remote

THING 5:

GIT HELPS YOU COLLABORATE

Committing helps you tell other people the story of your project

Remotes mean other people can access your project

Merges help manage combining your work with someone else's

Git allows lots of people to work on the same project, which is why people suffer through the terrible UX of it.

JARGON

Git terms we've covered

repository	your project folder
commit	a snapshot of your repo
hash	an id for a commit
checkout	time travel to a specific commit
branch	a movable label that points to a commit
merge	combining two branches
remote	a computer with the repository on it
clone	get the repository from the remote for the first time
push	send commits to a remote
pull	get commits from a remote

1. Tell the story of your project
2. Travel back in time
3. Experiment with changes
4. Back up your work
5. Collaborate on projects