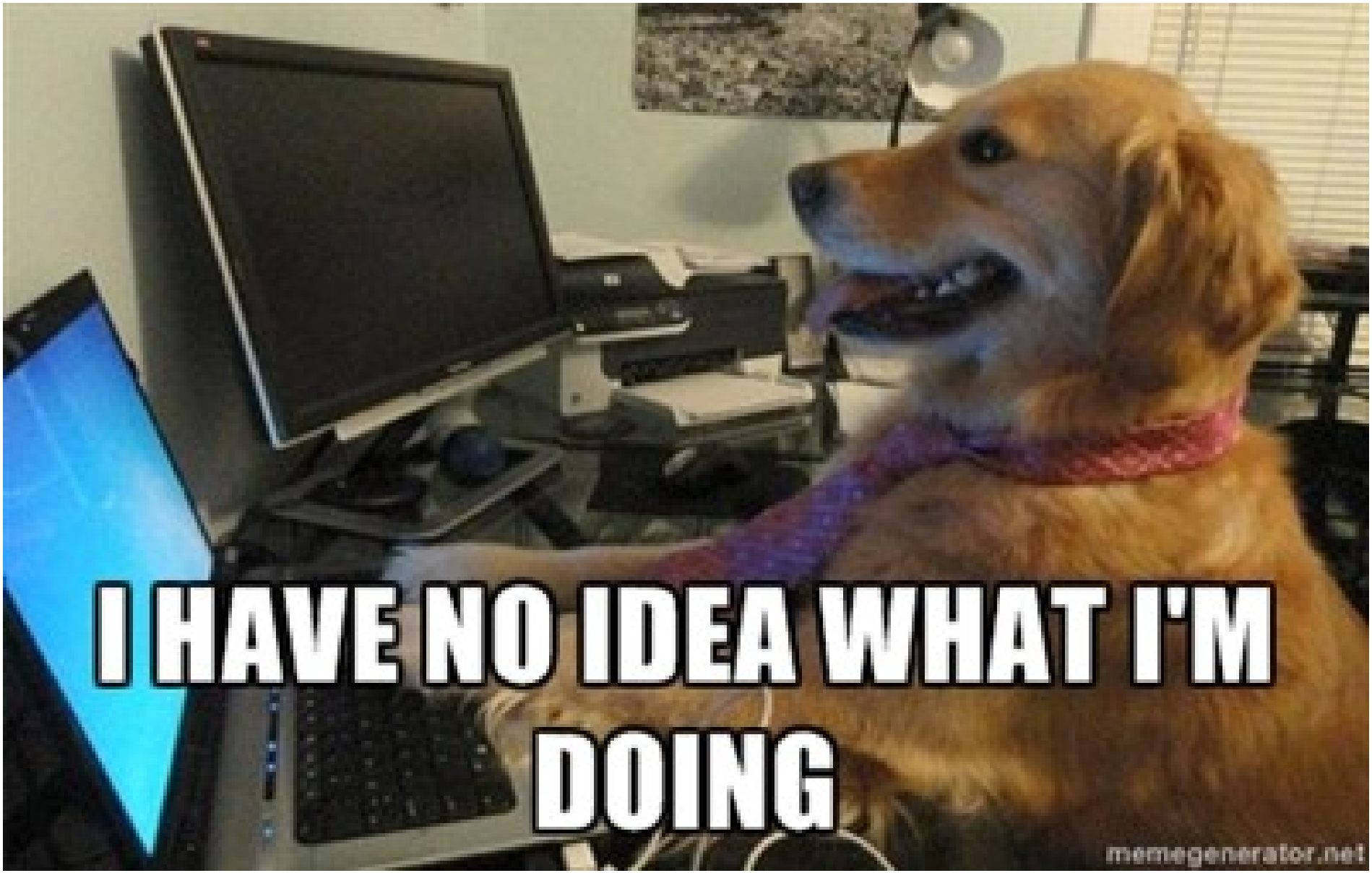
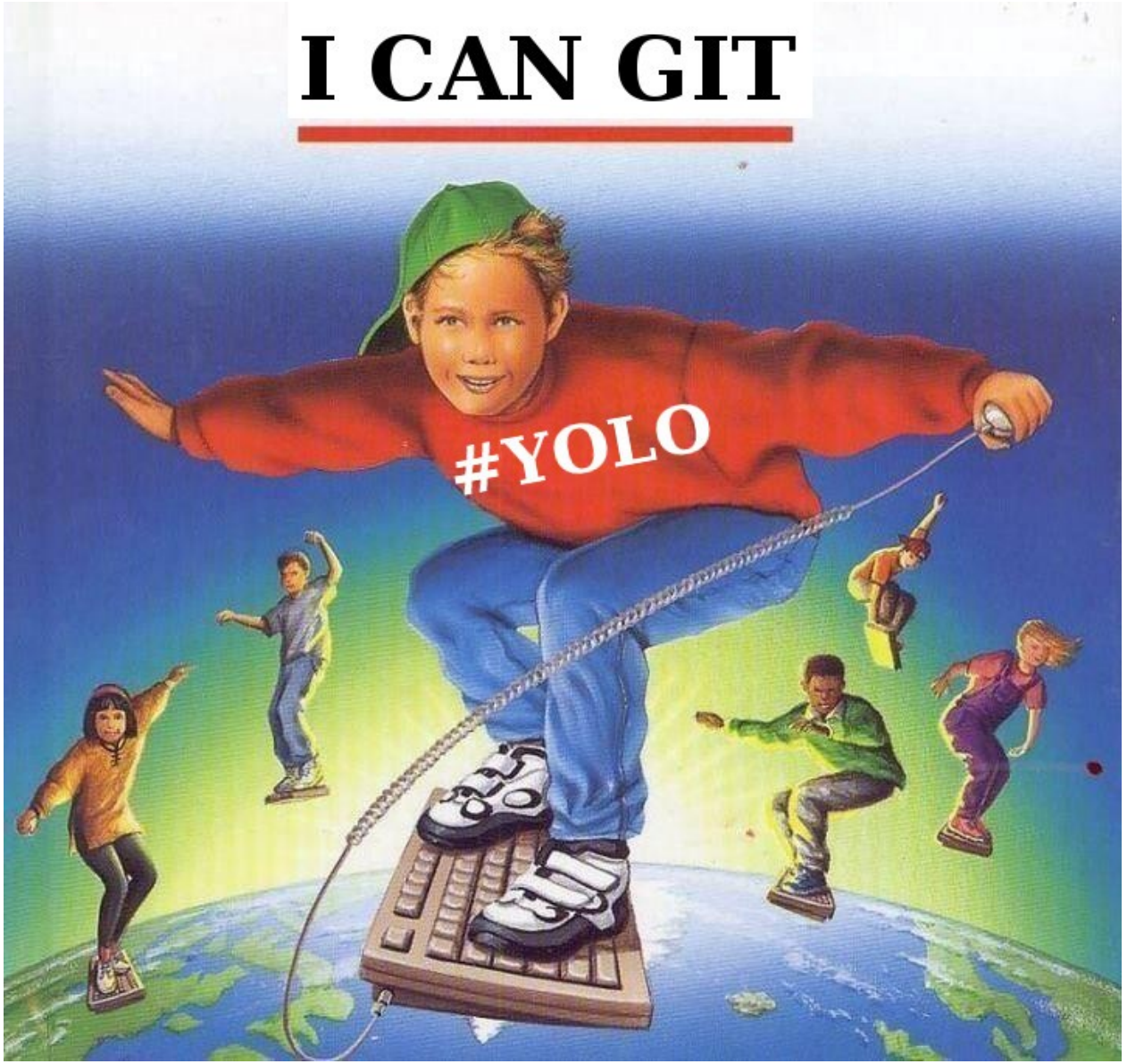
GIT workshop @ascribe

FROM THIS



TO THIS

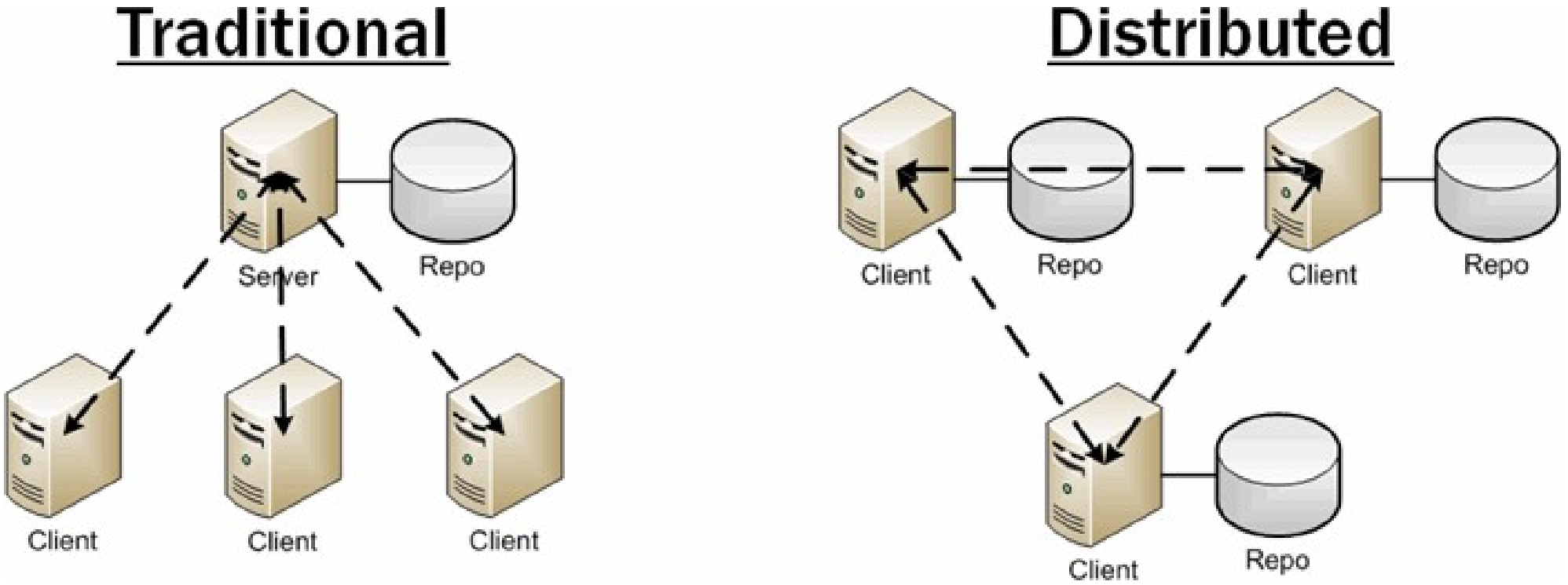


What is a VCS

VCS stands for Version Control System.

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

Git is a distributed VCS



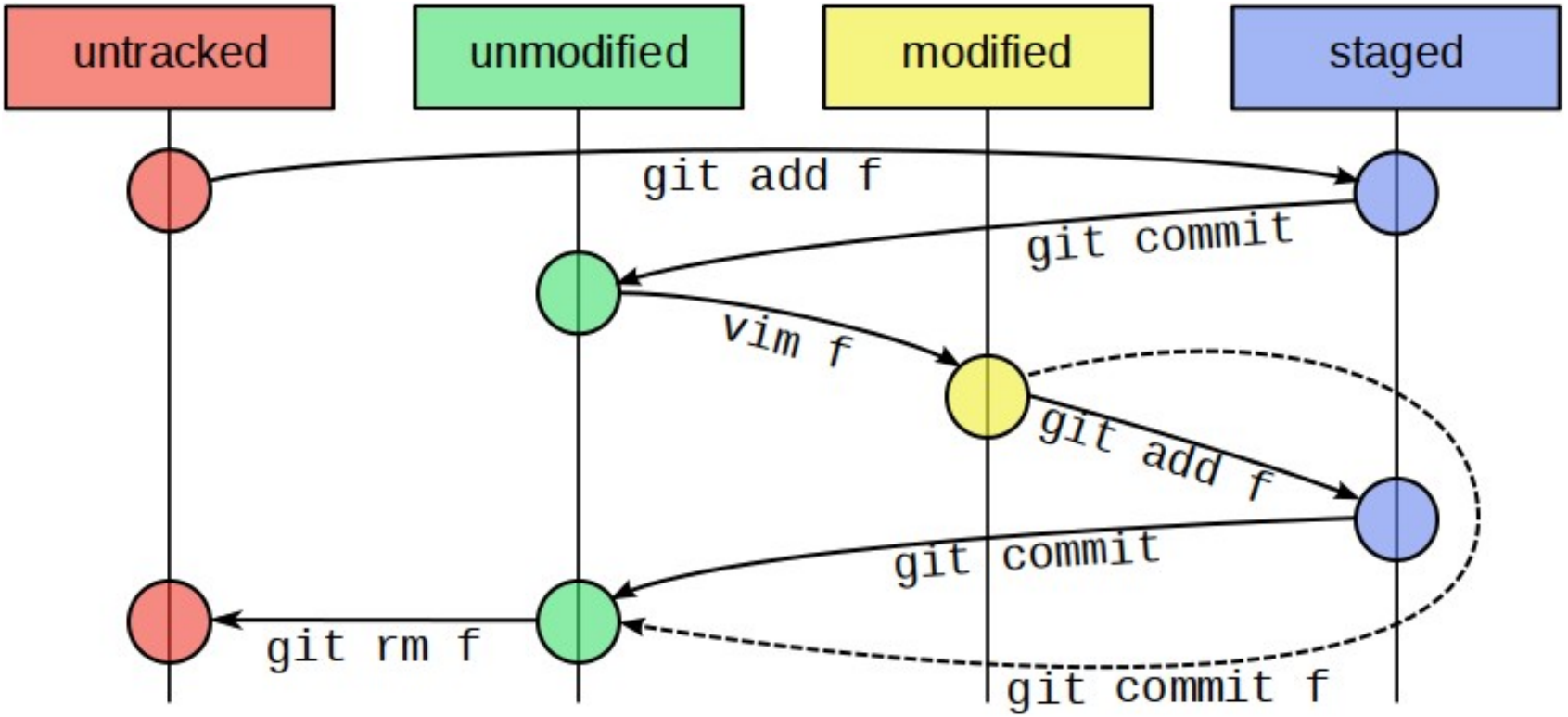
Instead of a central repository (SVN/CVS) you can have multiple repositories talking to each other.

You can use Git in your local filesystem, with no need to push your changes online. (nope, this was not possible before)

A repository in Git is called **REMOTE**

[It's a Acyclic Directed Graph!]

Git — status of a file



[It's a finite state automaton!]

https://github.com/git/git-scm.com/issues/288

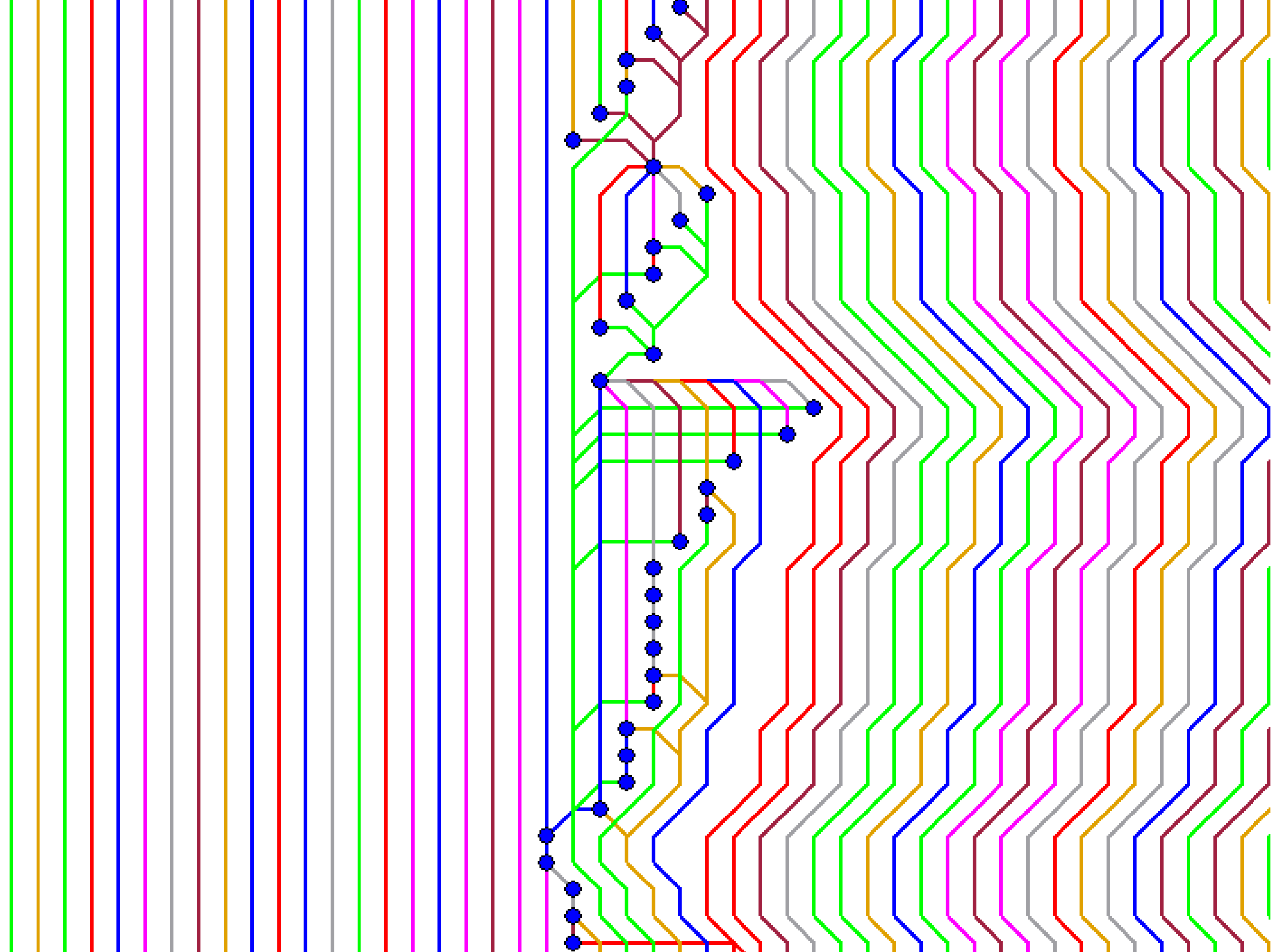
live demo

Interlude: how to write a commit message

- Use the present tense, no punctuation, no full stop at the end, and <72 char
- you write it once, you read it thousand of times: think about your future self reading the message again.
- You can try to write the commit message BEFORE changing the code

More here:
http://chris.beams.io/posts/git-commit/

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.



Git — branches

- Branches are a way to group a set of related commits
- Branches represent a feature, a fix, or an improvement
- Branches keep the codebase clean
- Branches allow to review the code before merging
- Branches are **lightweight**!

BRANCH ALL THE THINGS!

A branch is nothing more than a pointer to a commit

live demo

Albi works on a new issue

```
# move to master
git checkout master

# update master to the latest version
git pull

# create a new LOCAL branch
git branch feat/more-cats-on-the-home-page

# switch to that branch
git checkout feat/more-cats-on-the-home-page
```

Albi adds some code

```
# ...coding coding coding...
vim templates/home.html

# add the changes to the staging area
git add templates/home.html

# create a meaningful commit message
git commit -m 'Add two more cats to the landing page'

# My local Git doesn't know where to push my commits in origin.
# As a convention, I use the same name I use locally.
# You need to do it just one time.
git branch --set-upstream-to=\
            origin/feat/more-cats-on-the-home-page\
            feat/more-cats-on-the-home-page

# and then finally push
git push
```

Albi asks Dimi to check the code with a *pull request**

```
# Hi there, this is Dimi, and I'm working on another feature
# on another branch. I still have some changes to commit
# but I want to wait, so I stash my changes to a temporary
# location
git stash

# Now my working directory is clean, all my changes have been
# moved to a temporary location.
# I need to sync my local repository with the remote one
git fetch

# I'm in sync, and I have Albi's changes in my local repository.
# I move my HEAD to the new changes
git checkout origin/feat/more-cats-on-the-home-page

# I'm in a “detached head” state, nothing wrong with that.
# It works fine if I want to quickly check out the code
```

** Albi opens a pull request using the BitBucket web interface*

Dimi wants to change Albi's code

```
# I cannot change stuff directly, I need to give a local name
# to Albi's branch before.
git checkout -b feat/more-cats-on-the-home-page

# OK, now I can add that “lil bub” image that is missing
vim templates/home.html

# Add my changes to the staging area
git add templates/home.html

# create a meaningful commit message
git commit -m 'Add lil bub to the landing page to improve sales'

# Again, I need to tell git where to put my commits.
# As a convention, I use the same name I use locally.
# You need to do it just one time.
git branch --set-upstream-to=\
             origin/feat/more-cats-on-the-home-page
             feat/more-cats-on-the-home-page

# and finally push
git push

# then I want to go back to my work
git checkout my-old-branch
git stash pop
```

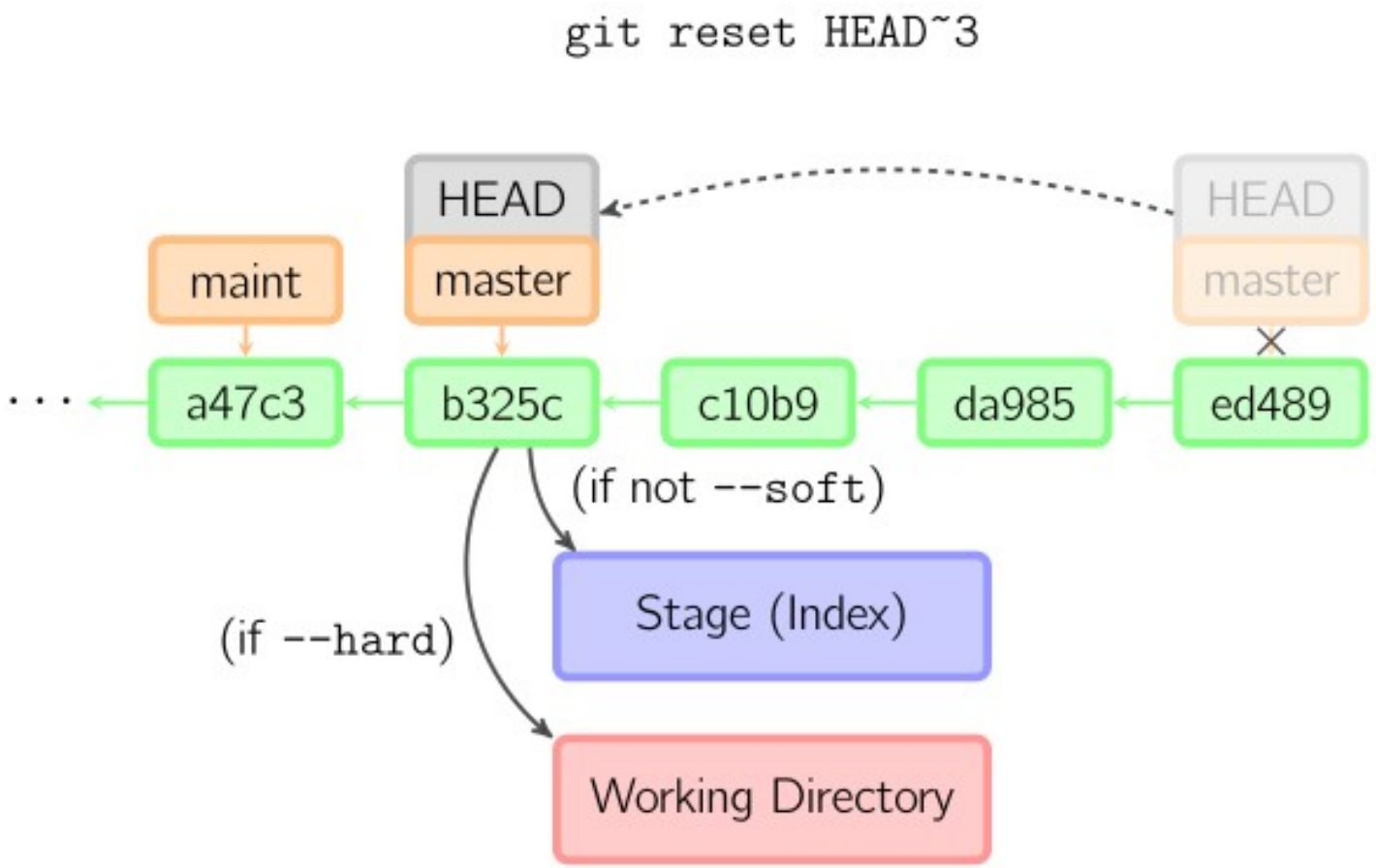
Albi wants to integrate Dimi's changes in master

```
# Albi pulls Dimi's changes
git pull

# and wants to integrate them in master.
git checkout master

# Albi syncs master because.
git pull

# and merges
git merge feat/more-cats-on-the-home-page

# then Albi pushes to two different remotes, the default one
git push

# and the production one
git push live master
```

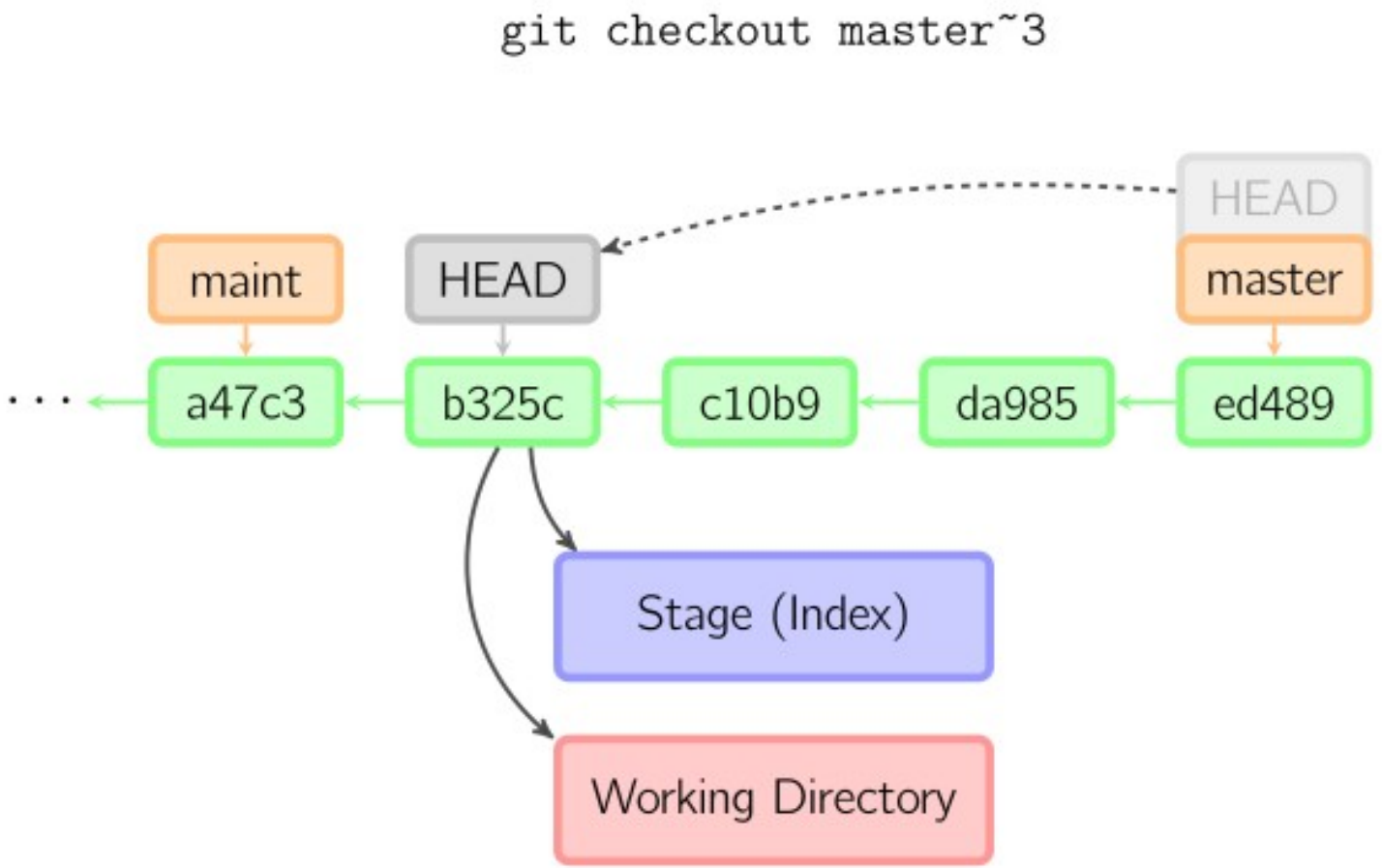
Great job!
lil bub is in the landing page!

Playing with the history

Git reset


git reset HEAD~3
HEAD
master
HEAD
master
maint
...
a47c3
b325c
c10b9
da985
ed489
(if not --soft)
Stage (Index)
(if --hard)
Working Directory

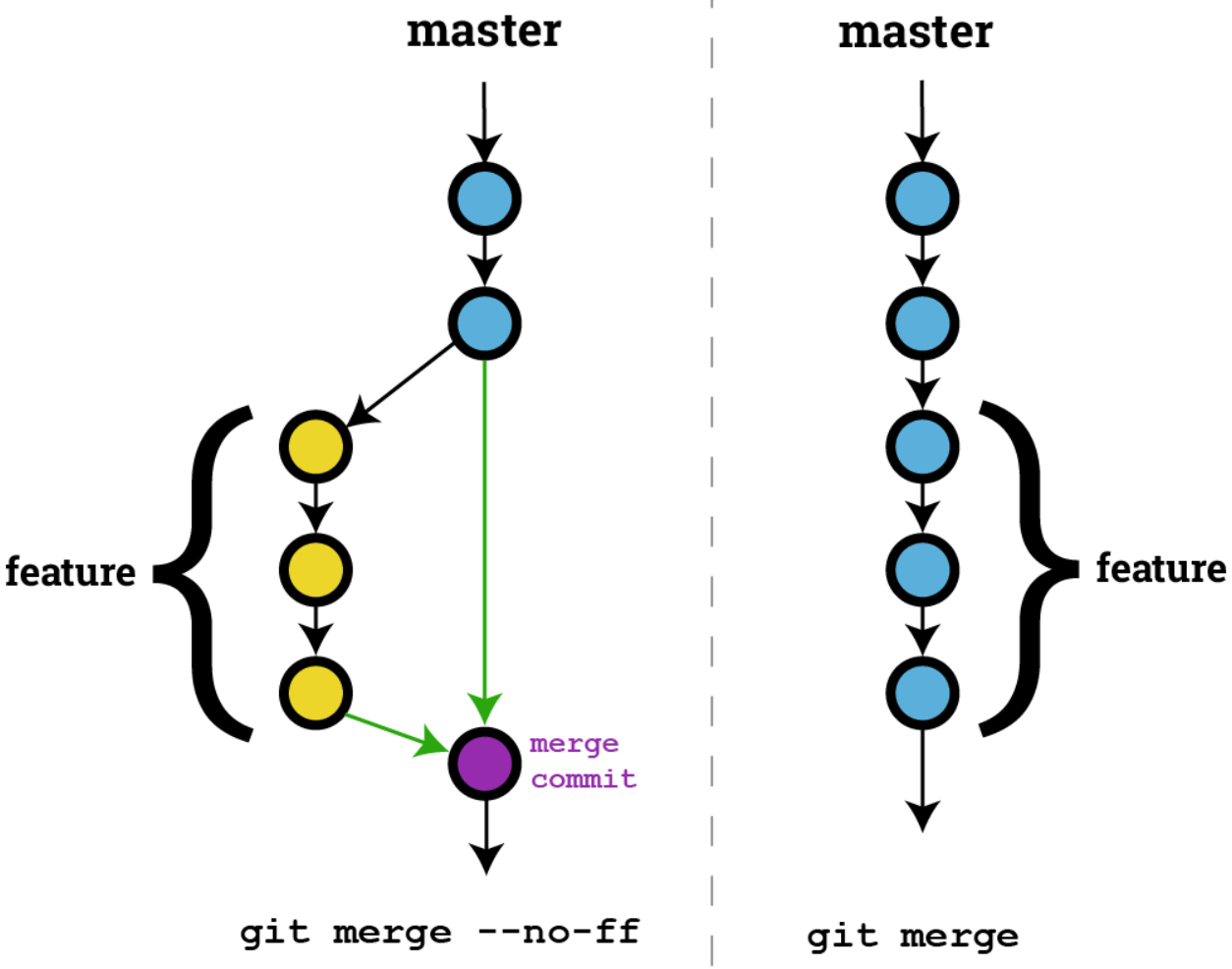
Git checkout



http://stackoverflow.com/questions/3639342/whats-the-difference-between-git-reset-and-git-checkout

live demo

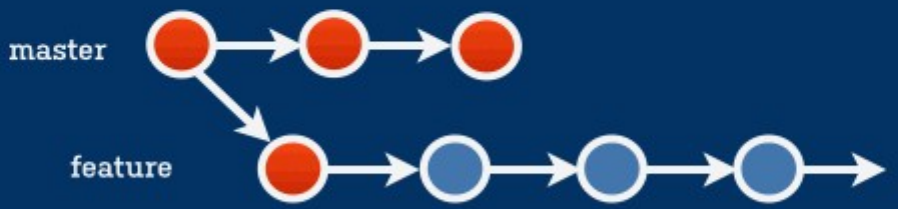
git merge --no-ff



git rebase

What is a **rebase**?

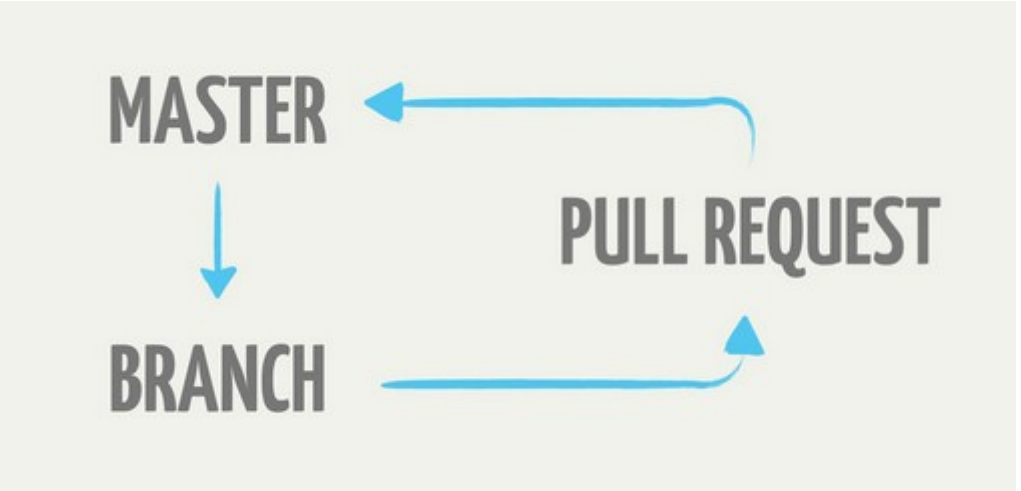
preserving the order
of change-sets



git rebase

TL;DR don't do it.

Github flow



see: https://guides.github.com/introduction/flow/

Extra demos

git merge --no-ff

aliases

watchtree

My personal advice

- Play a lot with a local repo and push your changes on a remote repo, BitBucket is for free
- If you are stuck, go to stack overflow and try to **understand** the answer
- Git man pages are **absurdly confusing**, so **absurdly confusing** that this* happened
- read a bit of this http://git-scm.com/book/en/v2 every night before sleeping

  and remember, Git is --hard, but it's also a lot of fun

* http://git-man-page-generator.lokaltog.net/

chi wai lau
@tabqwerty

Follow

"git gets easier once you get the basic idea that branches are homeomorphic endofunctors mapping submanifolds of a Hilbert space."

RETWEETS 1,841 FAVORITES 609

11:28 PM - 9 Mar 2011