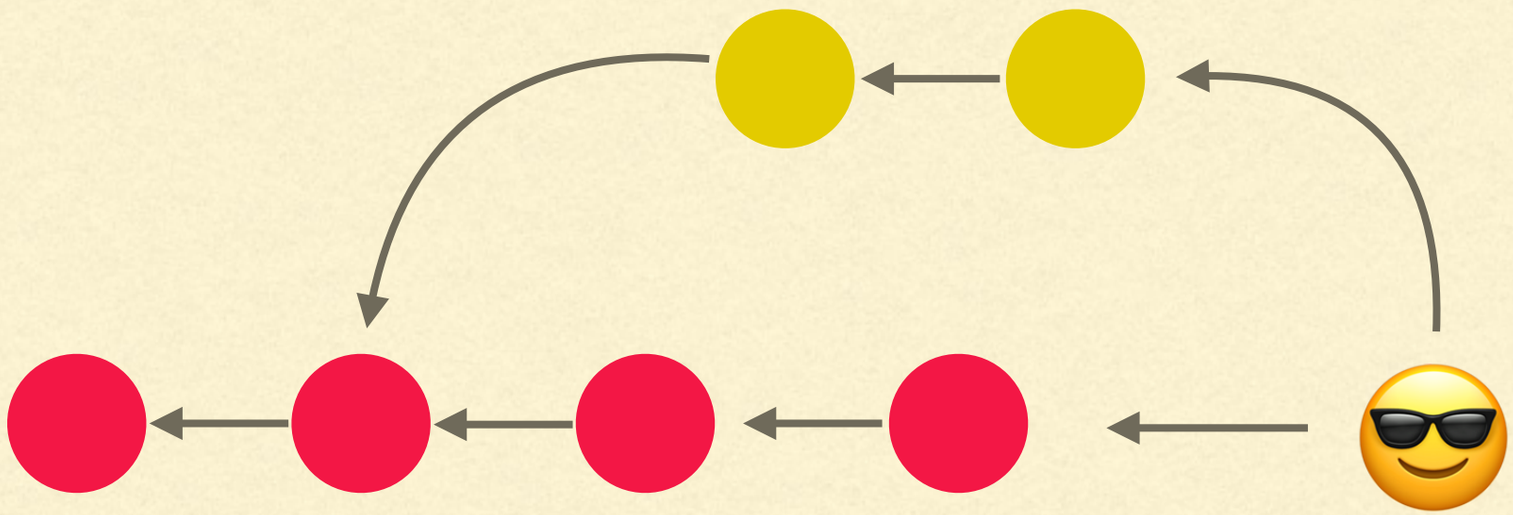
JMB, 10/4/19

Github Intro



$ git init

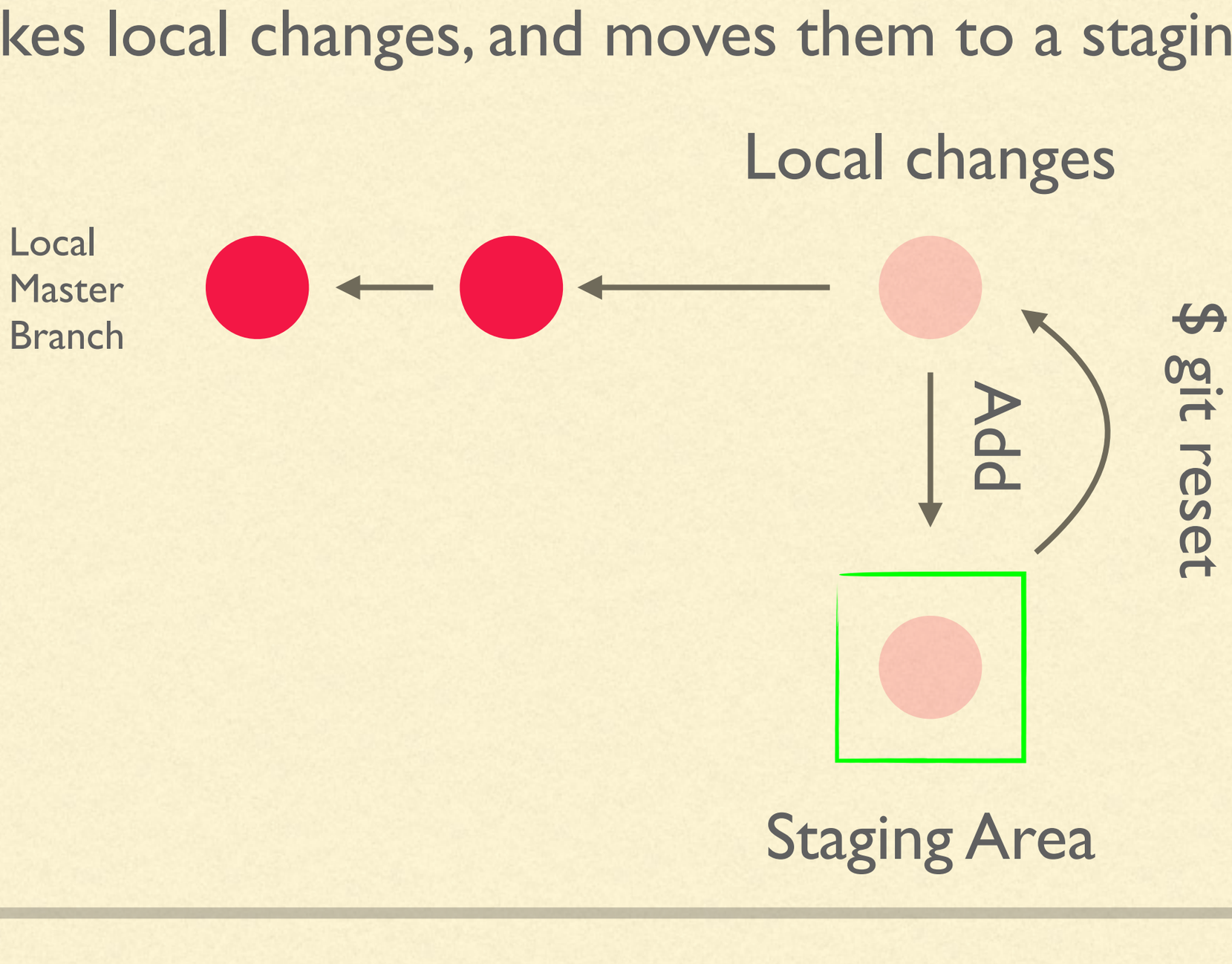
- Let's start local: we are in a local directory
- Git init changes a directory into a repository
    - Adds a .git subdirectory
    - . git is invisible but can be viewed with ls -a
- If you have nested git repositories, remove the inner .git folders
    - $ rm -rf .git

$ git init note

- Note: immediately after init, there are no branches, not even a master branch.
- $ git branch
  - returns nothing
- Master branch comes into being with the first commit.

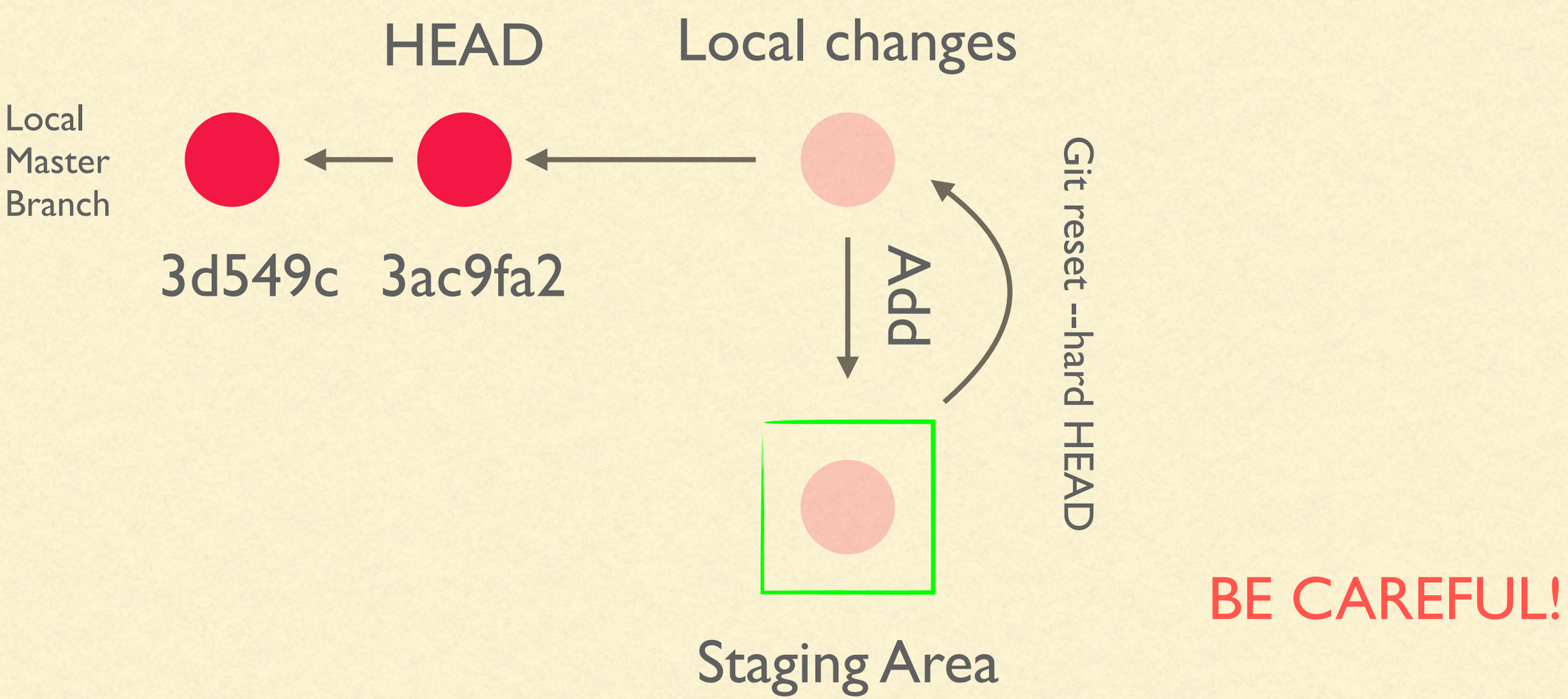
$ git add

- Takes local changes, and moves them to a staging area (index)



If you want to revert to a previous commit…

- Git reset --hard will delete your local changes

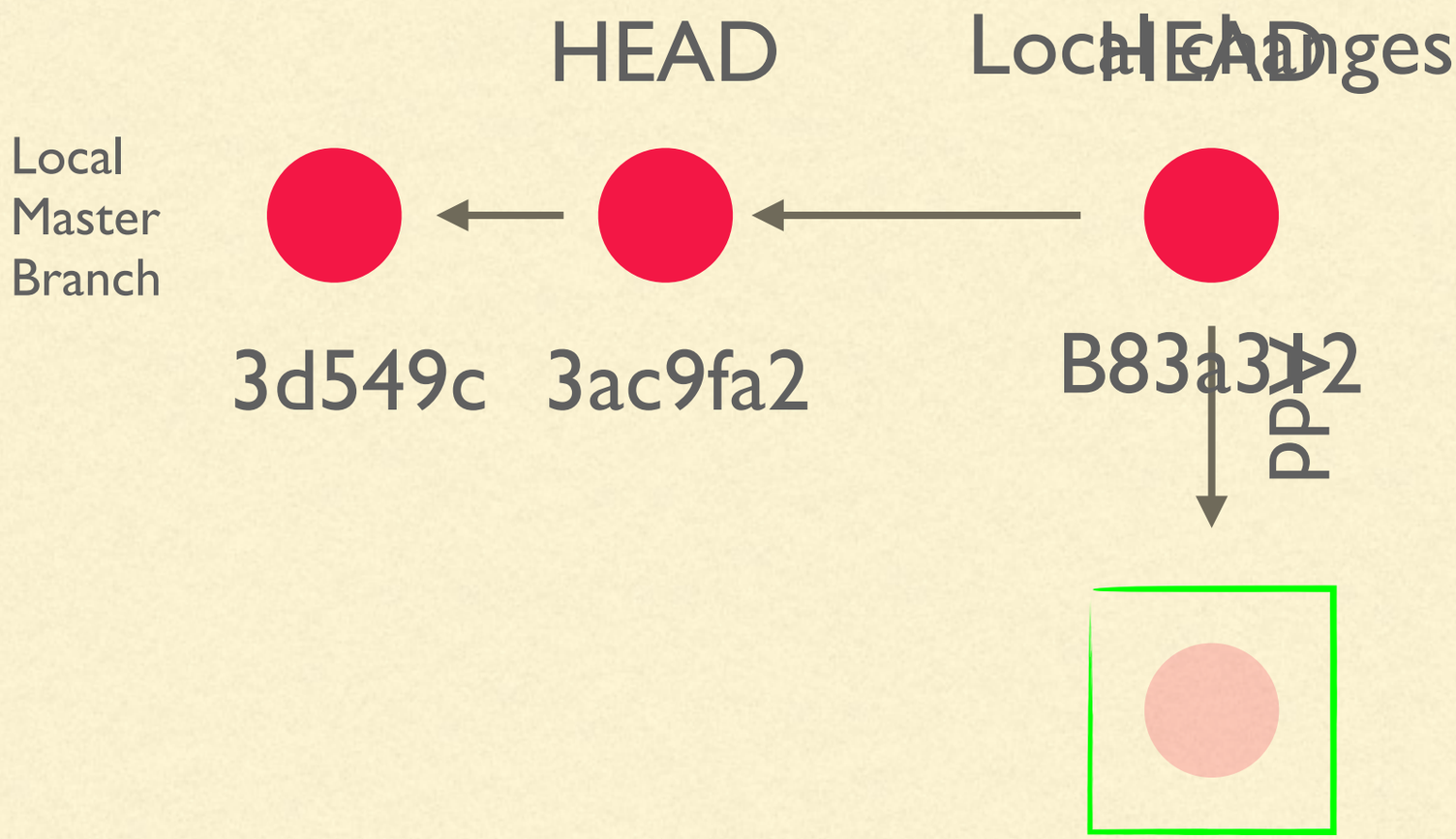


BE CAREFUL!

$ git add …

- $ git add <file_name>
- $ git add <file_name> <file_name_2>
- $ git add -A (stages all files, including removed)
- $ git rm <file_name> (stages a specific deleted file)

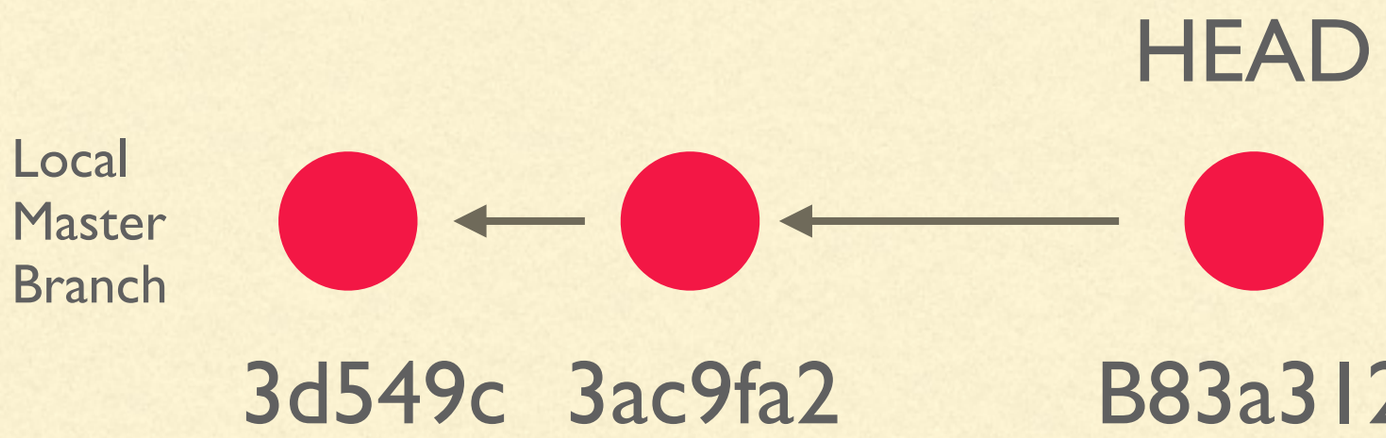
Commits



Staging Area

$ git commit -m “Add meaningful message”

Parents and HEAD



Every commit, excluding the first, points back to at least one parent.

The HEAD points to the most recent commit of the current branch.

HEAD can be referenced recursively: HEAD, HEAD^, HEAD^^

Commits

- Commits have a sha-1 (secure hash algorithm 1) to identify it

60af1dc41759ddb97ae8adf0d593c6ec1f15a335

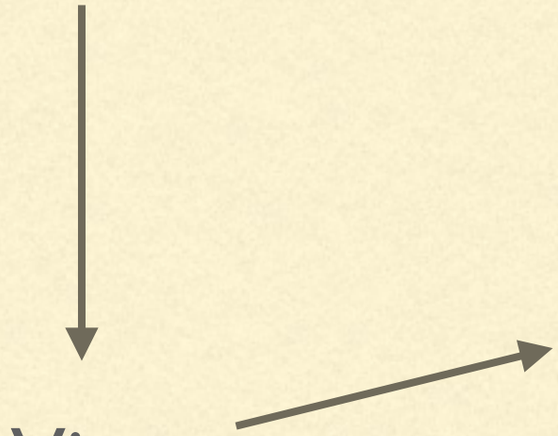
Commit Conventions

$ git commit -m “Put a message here”

- Must be less than 50 characters
- Capitalize but no period
- Imperative mood

Longer messages

$ git commit



Vim

Put a message here
(space)
Then go ahead and type
A longer description of
changes here

Ways to view commits

$ git log

$ git log --since='yesterday'

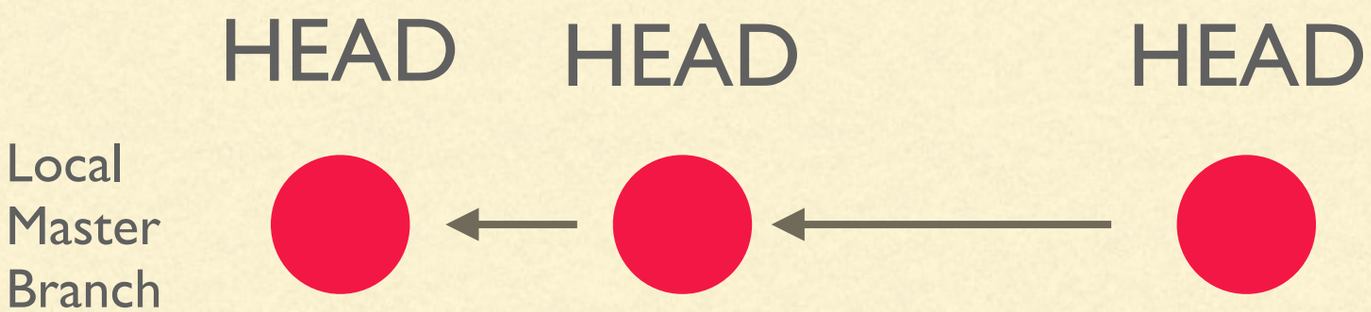
$ git log --since='9/30/19'

$ git log --diff-filter='M'

$ git log --diff-filter='D'

$ git log <filename>

What if you want to see the state of the repo at previous commit?



$ git checkout 3ac9fa2 Or $ git checkout HEAD^

$ git checkout 3d549c Or $ git checkout HEAD^^

You are in 'detached HEAD' state.

Detached HEAD state is fine

- Look around at the state of the repo
- When you run git branch -v, you see:

> (HEAD detached at b83a3fa) b83a3fa changed to …

- If you checkout your master branch, the current state of your repo will reappear.

Branches

Branch Workflow

$ git checkout -b new_branch *

$ git add <file_name>

$ git commit -m “meaningful message”

$ git checkout master

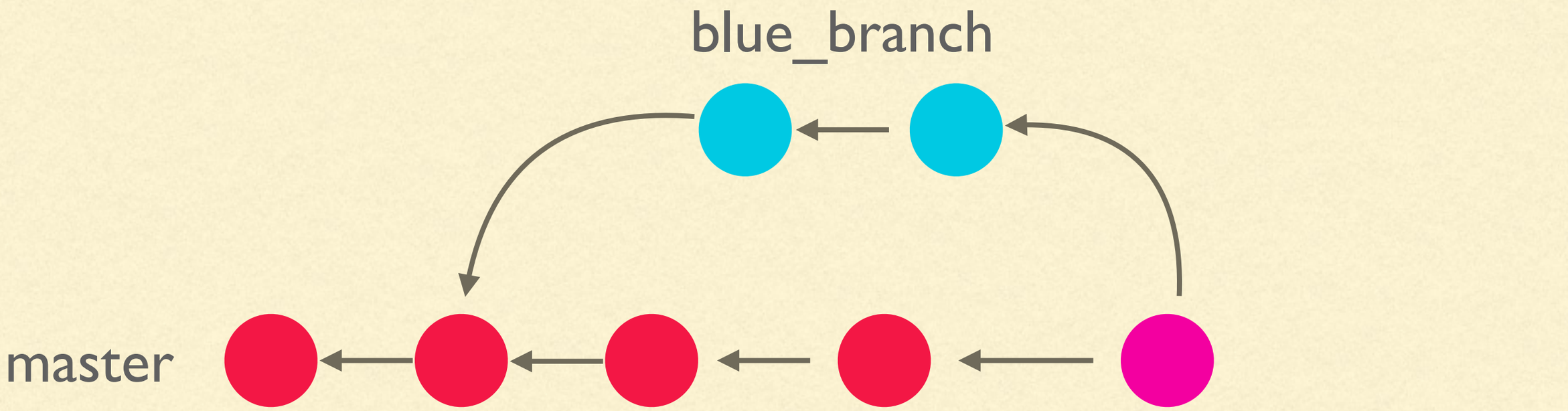
$ git merge new_branch

$ git branch -d new_branch

(git will warn if there are unmerged changes)

* Assumes no changes needing a commit

Branches

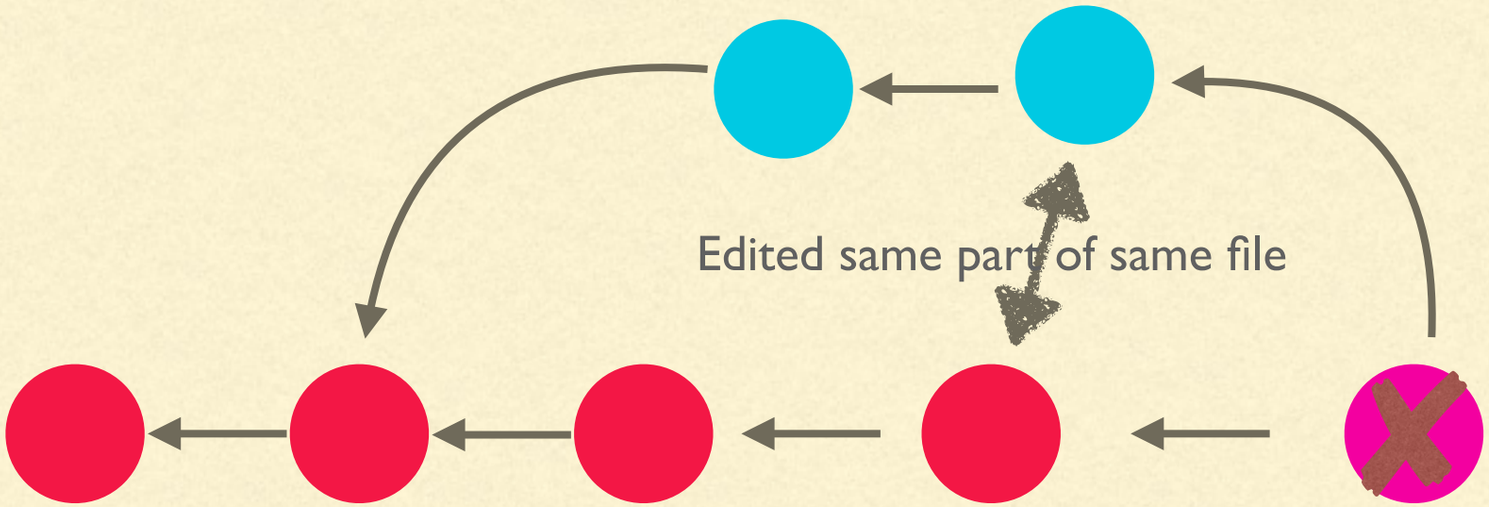


$ git checkout -b blue_branch

$ git merge blue_branch

$ git branch -d blue_branch
(but can still checkout blue branch commits)

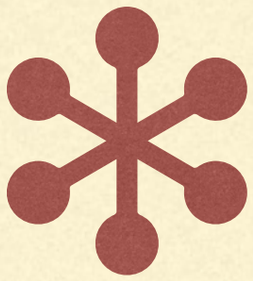
Branches - merge conflict



Branches - merge conflict

```
<<<<<<< HEAD
Some text
=======
Some different text
>>>>>>> other_branch
```

Fix conflicts (i.e. go into the file and make it look how you want it)
Then add, commit



Checking out new branch does not work

- Commit changes first

- Or if there are too many changed files
  - Git stash
    - A quick fix, but not recommended to use often instead of committing

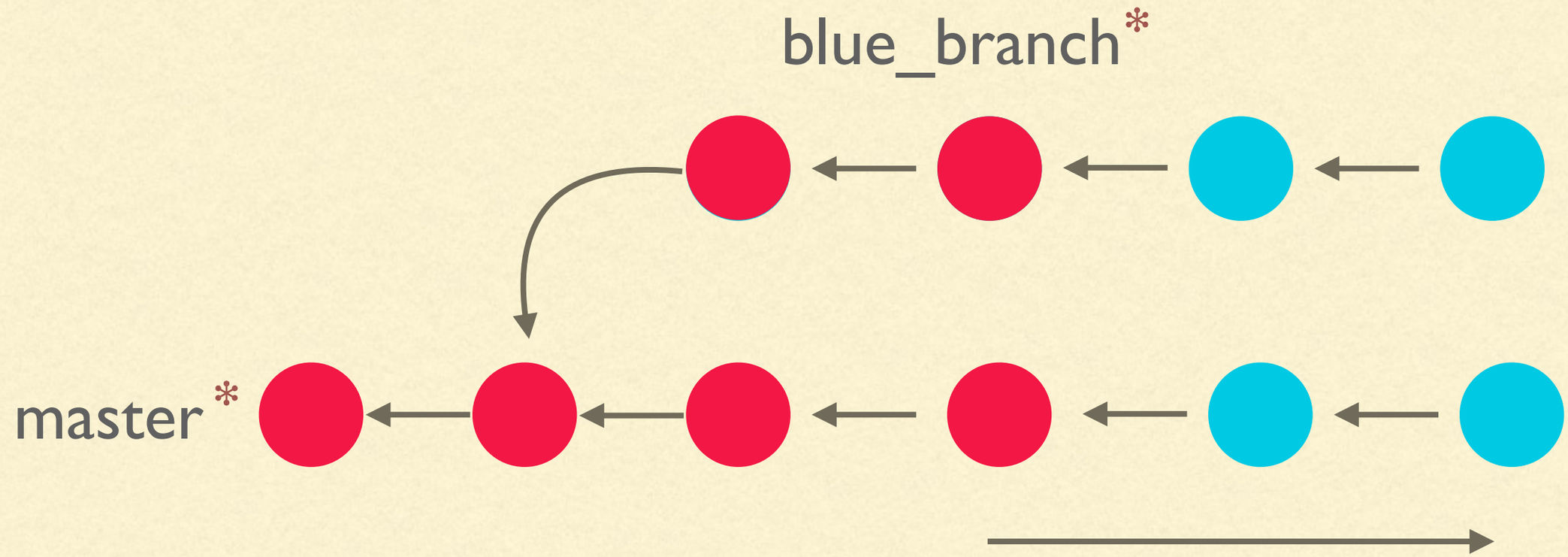
git 

```
$ git stash list

$ git stash show stash@{1}

$ git stash apply stash@{1}
```

Rebase



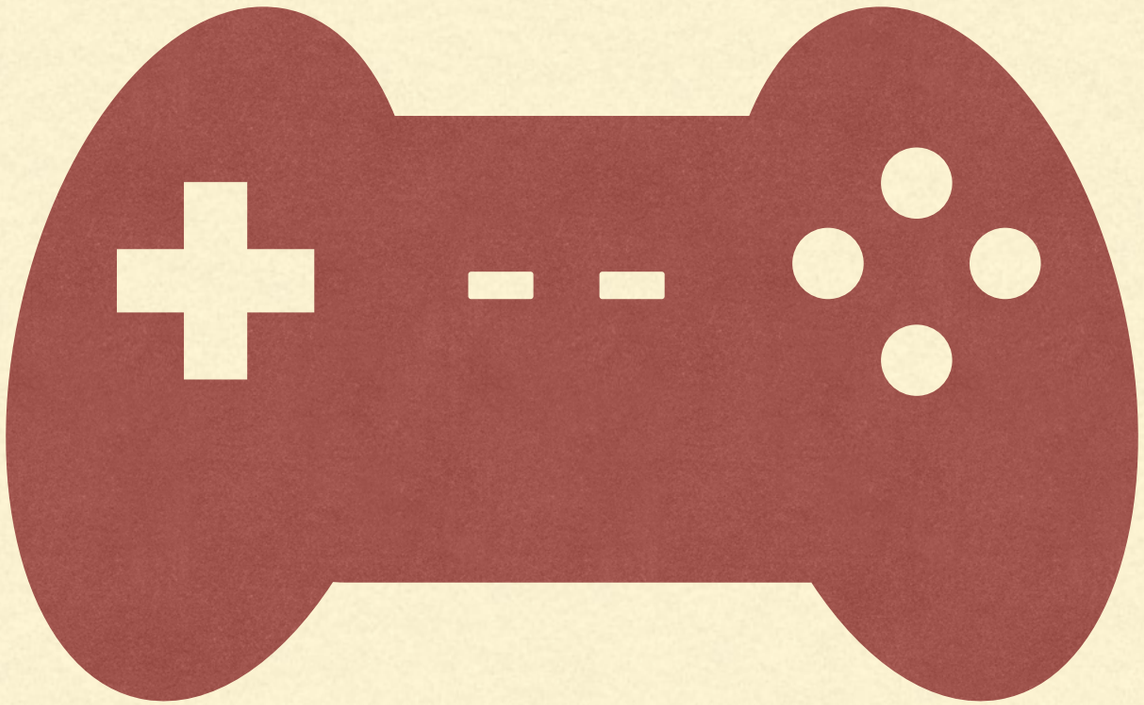
$ git rebase master $ git checkout master $git merge blue_branch

Master fast-forwards

Rebase - rules of 

- Don’t rebase commits which others could have incorporated into their work
- Rebase local changes you haven’t shared yet before you push.

Remote Repositories

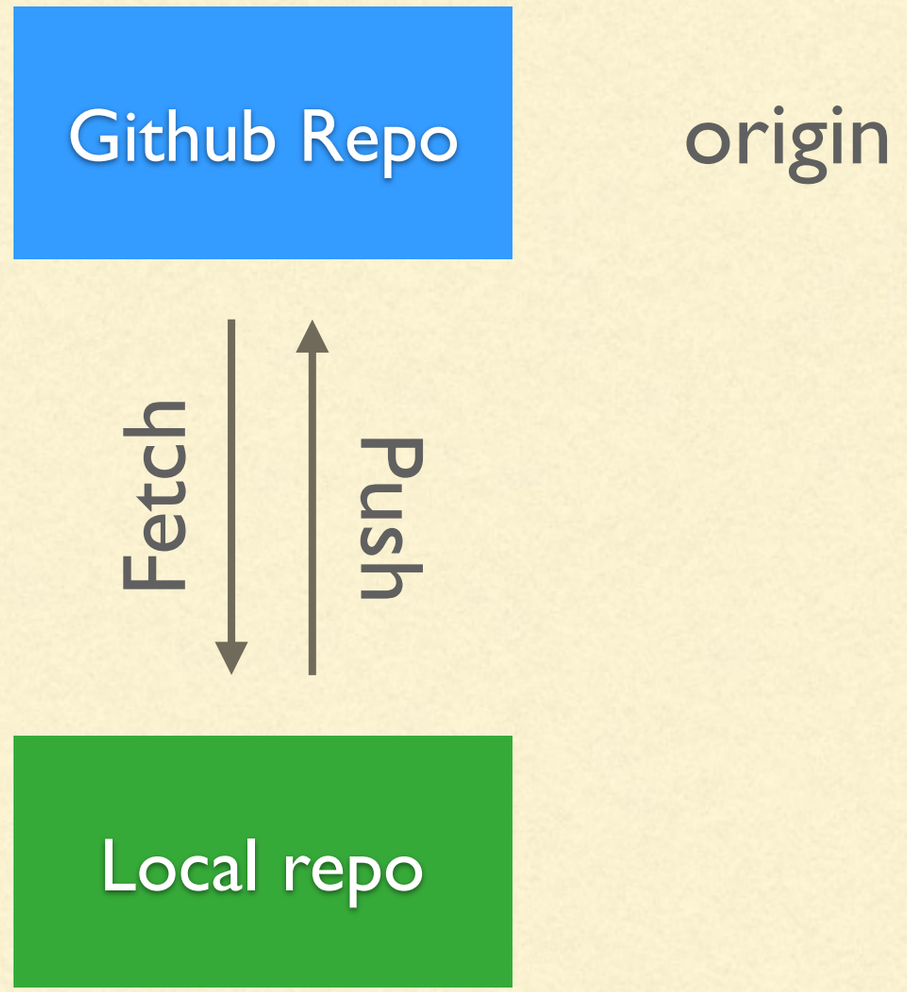


Git clone

- Git clone <url_of_github_repository>
  - Calls init
  - Then copies the remote repository files

The remote origin is automatically set to the cloned url
And the local master branch is associated with the origin master

Cloned repository structure



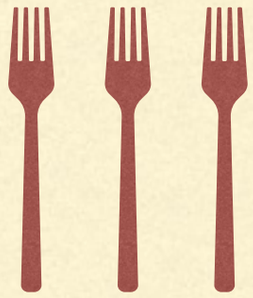
Remote - origin

- **git remote -v**

  $ origin https://github.com/courtneyb-123/lottasealice.git (fetch)
  $ origin https://github.com/courtneyb-123/lottasealice.git (push)

- Remote could have any name
- It defaults to origin because that is the url from which it was originally cloned

Forks



kurtv-456/lottasealice.git

courtneyb-123/lottasealice.git

origin

Forked repo

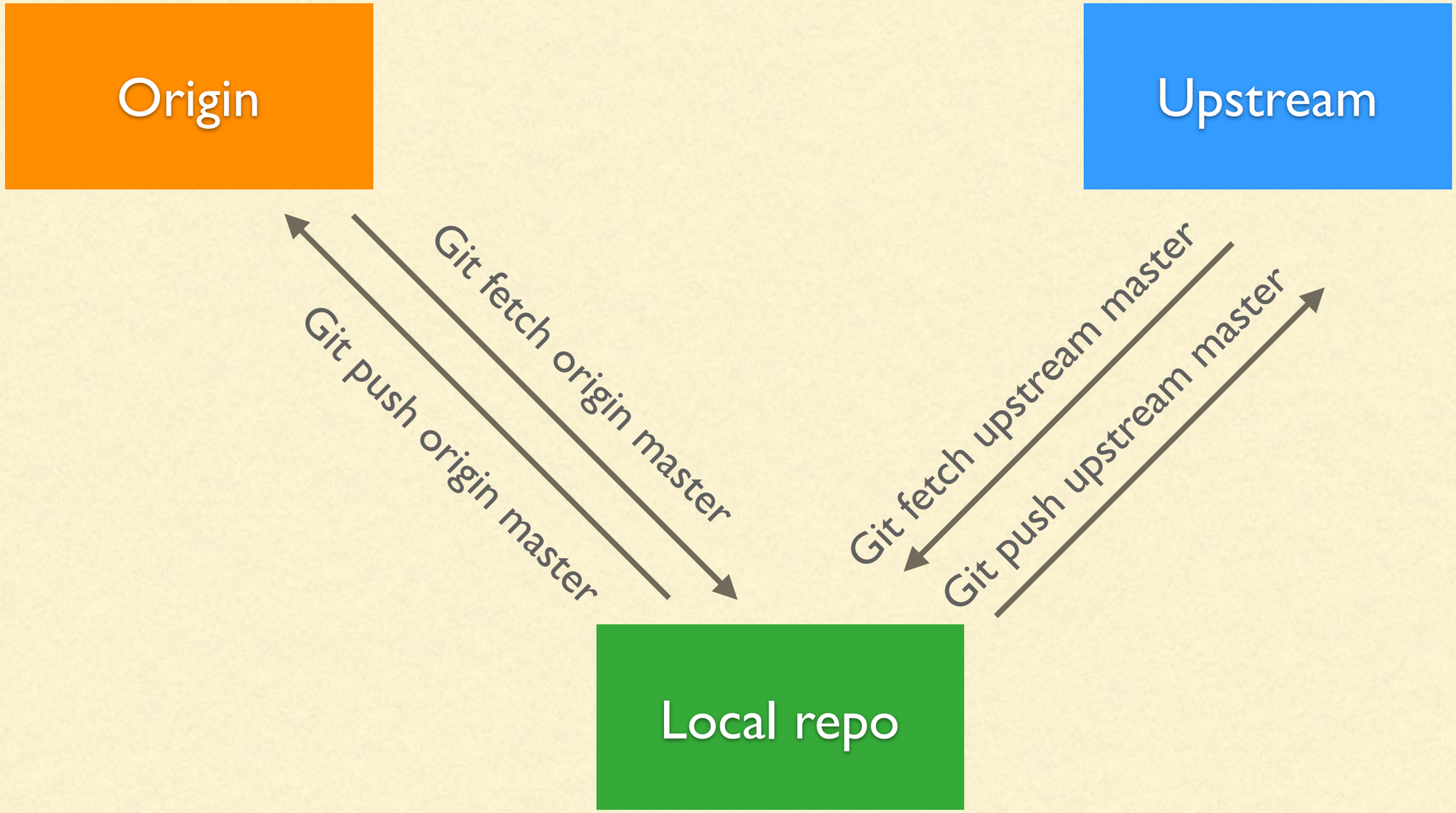
Github Repo

$git clone

Local repo

kurtv-456/lottasealice.git

courtneyb-123/lottasealice.git



$ git remote add upstream https://github.com/courtneyb-123/lottasealice.git

Remotes - upstream

$ git remote -v

$ origin https://github.com/kurtv-456/lottasealice.git (fetch)
$ origin https://github.com/kurtv-456//lottasealice.git (push)
$ upstream https://github.com/courtneyb-123/lottasealice.git (fetch)
$ upstream https://github.com/courtneyb-123/lottasealice.git (push)

Remotes - Just to be clear

remote

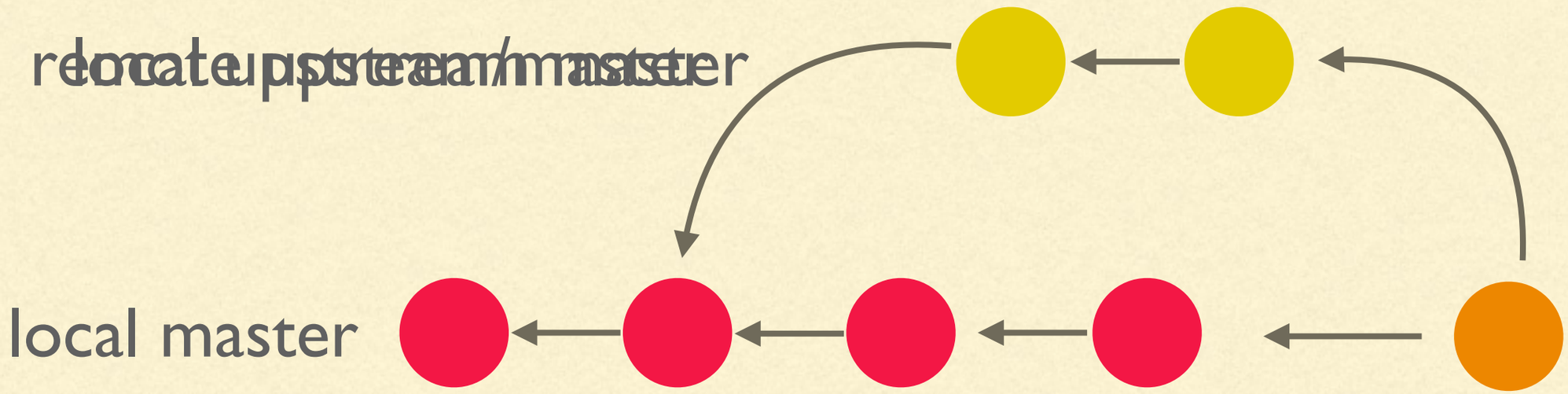
Git fetch upstream master

branch

Why could it be batch to fetch without branch?

All changes across branches will be pulled

Branches and forks



$ git fetch upstream master

What happens when you push to a remote and the branch doesn't exist?

$ git push origin <new_branch>

Git push --set-upstream origin <new_branch>

--set-upstream is saying "there is no <new_branch> in the repo you assigned to origin."

- All local branches must have a remote branch for a successful push.

$ git push upstream <new_branch>

Git push --set-upstream upstream <new_branch>

.gitignore

- Put it in the working directory
  - Not in the .git

- What to put in there
  - *.csv
  - *.ipynb_checkpoints
  - Passwords/api keys