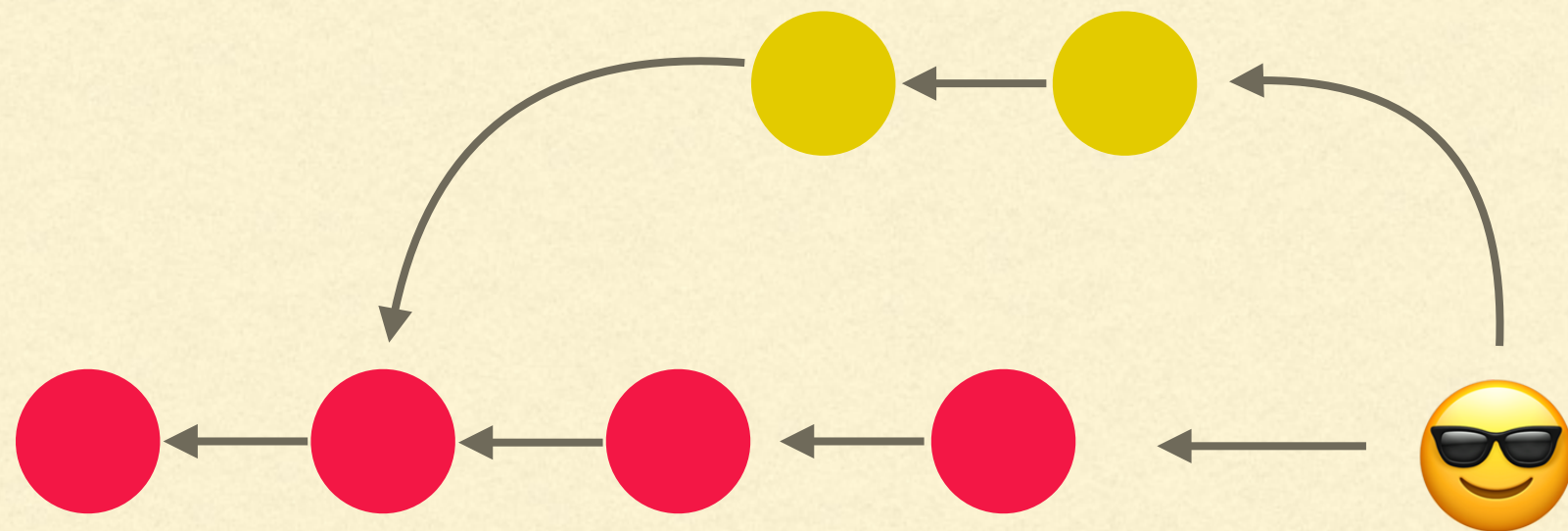


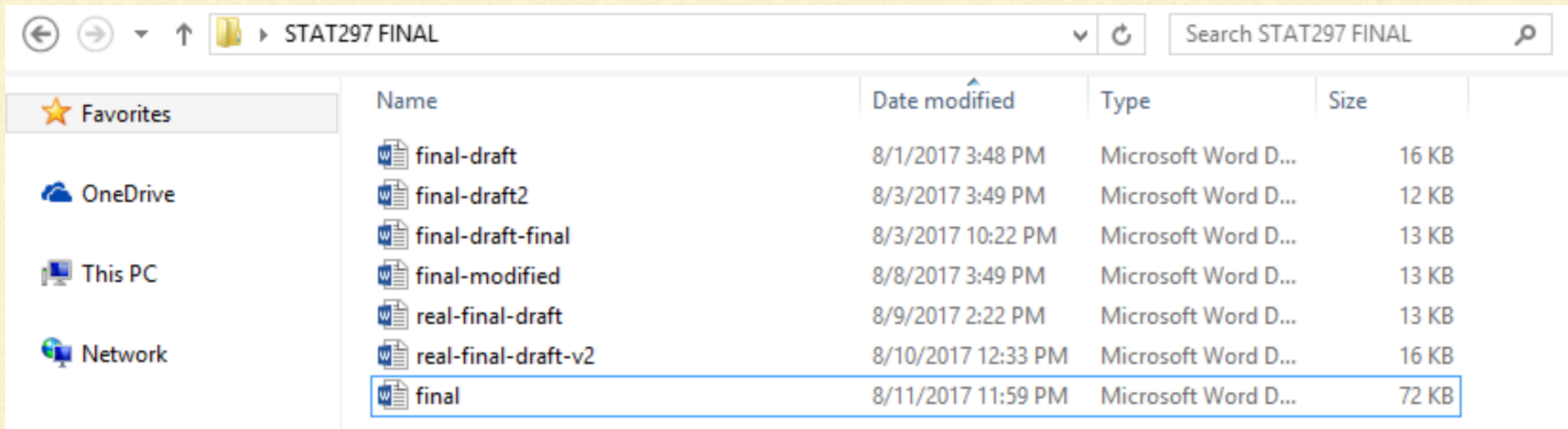
# Github Intro










---

# How many of you have seen a folder like this?

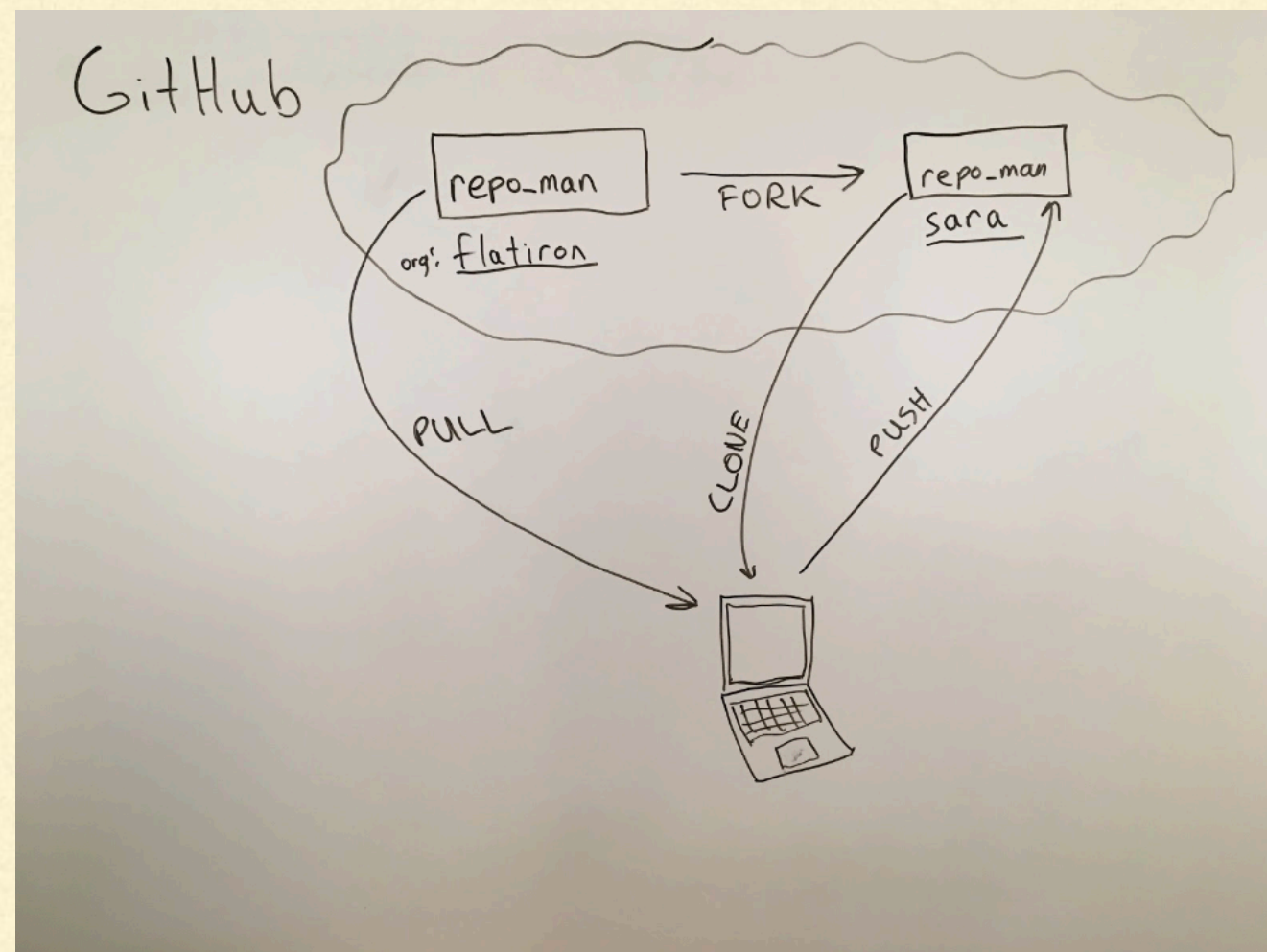
---



STAT297 FINAL				
Search STAT297 FINAL				
Name	Date modified	Type	Size	
 final-draft	8/1/2017 3:48 PM	Microsoft Word D...	16 KB	
 final-draft2	8/3/2017 3:49 PM	Microsoft Word D...	12 KB	
 final-draft-final	8/3/2017 10:22 PM	Microsoft Word D...	13 KB	
 final-modified	8/8/2017 3:49 PM	Microsoft Word D...	13 KB	
 real-final-draft	8/9/2017 2:22 PM	Microsoft Word D...	13 KB	
 real-final-draft-v2	8/10/2017 12:33 PM	Microsoft Word D...	16 KB	
 final	8/11/2017 11:59 PM	Microsoft Word D...	72 KB	



Version control is a system that records changes to a file or a set of files over time so that you can recall specific versions later.

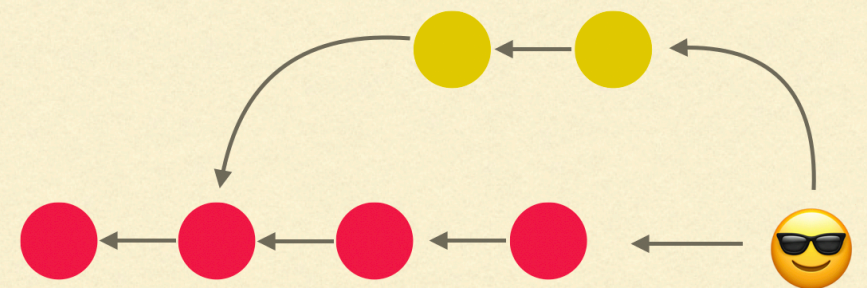




---

# Overview

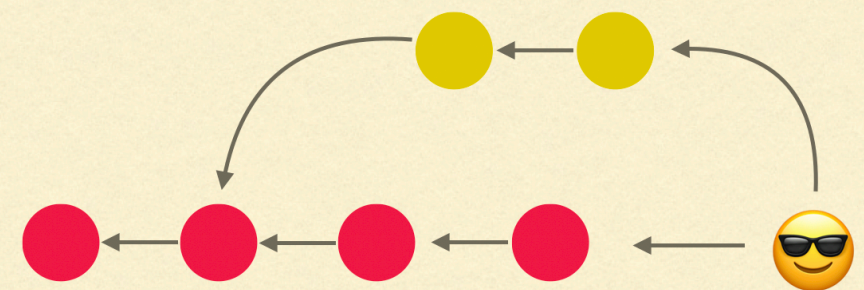
- Why Git
- Git basics and workflow (local)
- Branching
- Remote Repos





# Where will we use git?

- Lecture materials and warmups will be deployed at <https://github.com/learn-co-students/chicago-ds-012720>
- End of module projects will live in Github
- You will link to your Github on your resume. Your projects will be a main talking point in interviews, and will be reviewed by employers to assess your DS skills



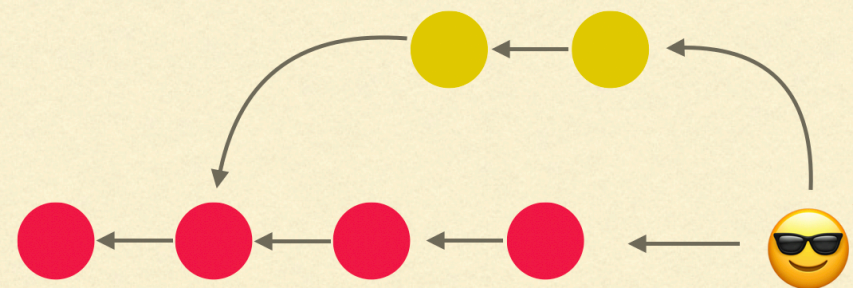


---

# Git basics - Local vs Cloud

---

- Git is a version control system which you can use without a connection to the internet. It is just a folder on your computer with special capabilities.
- Github, Bitbucket and the like are hosts which allow for developers shared access to projects for the purpose of collaboration.





---

# \$ Let's start local: git init

---

- Open a terminal window and navigate to your Documents directory (`cd ~/Documents`)
  - Make a new directory called `git_practice` (`mkdir git_practice`)
  - 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.
-

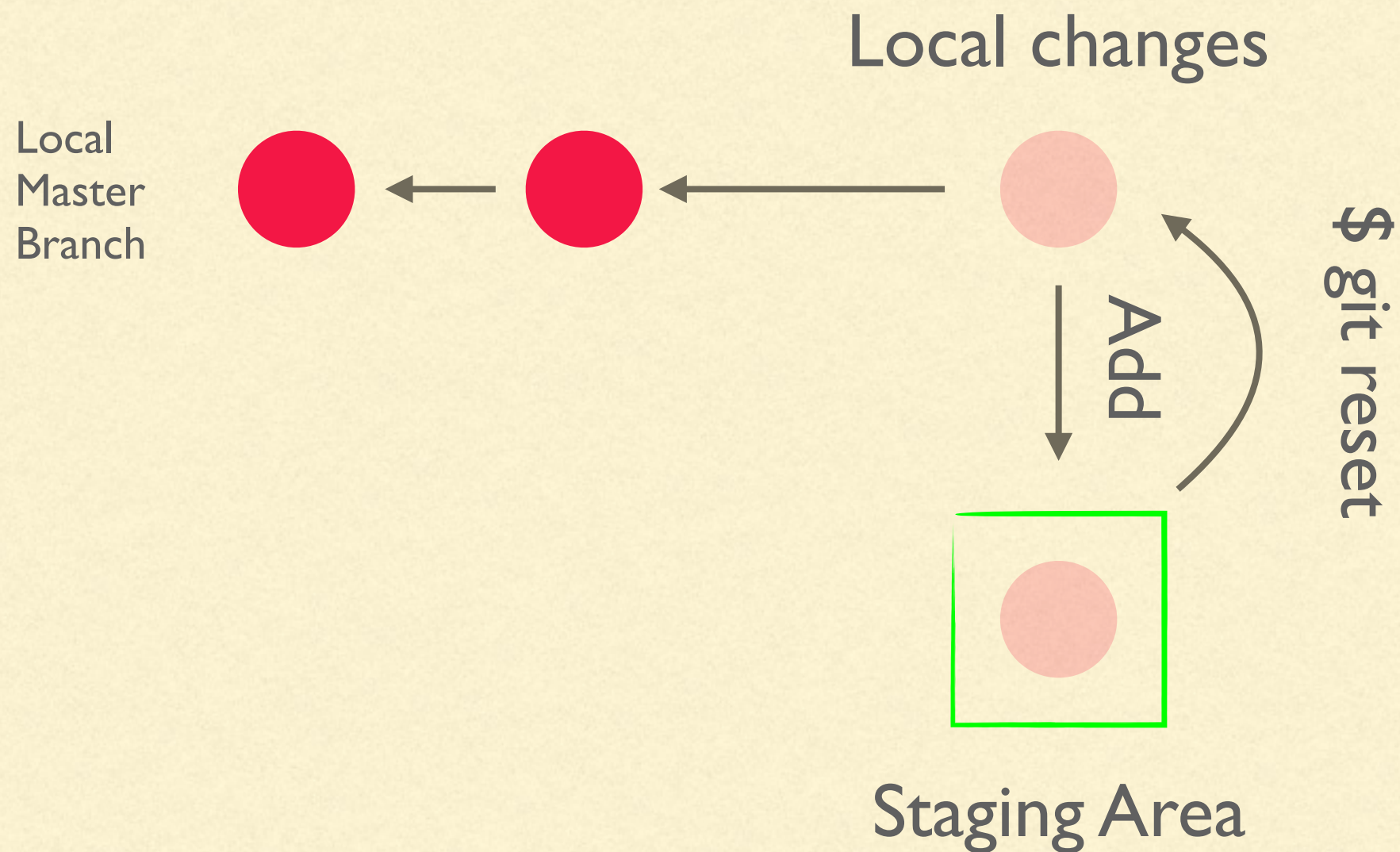


---

# \$ git add

---

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



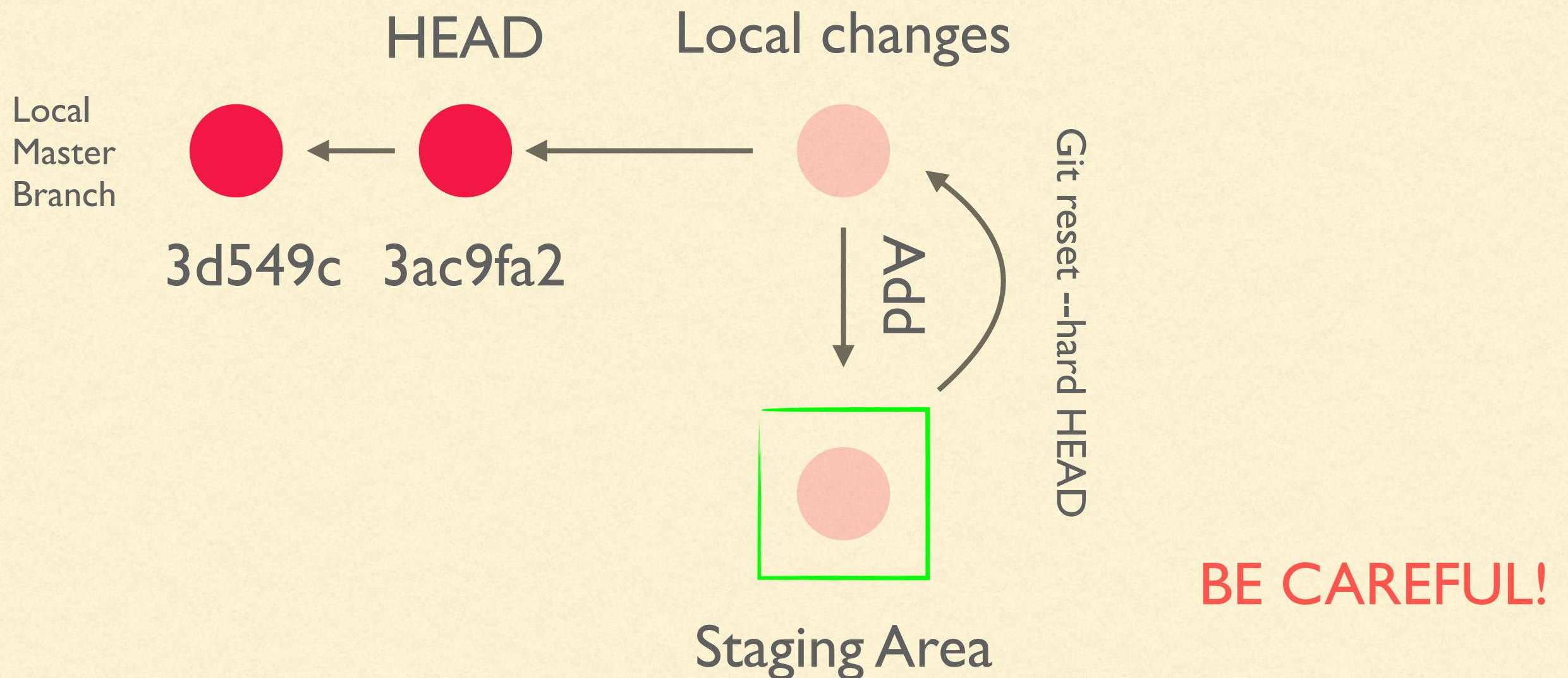


---

If you want to revert to a previous commit...

---

- Git reset --hard will delete your local changes





---

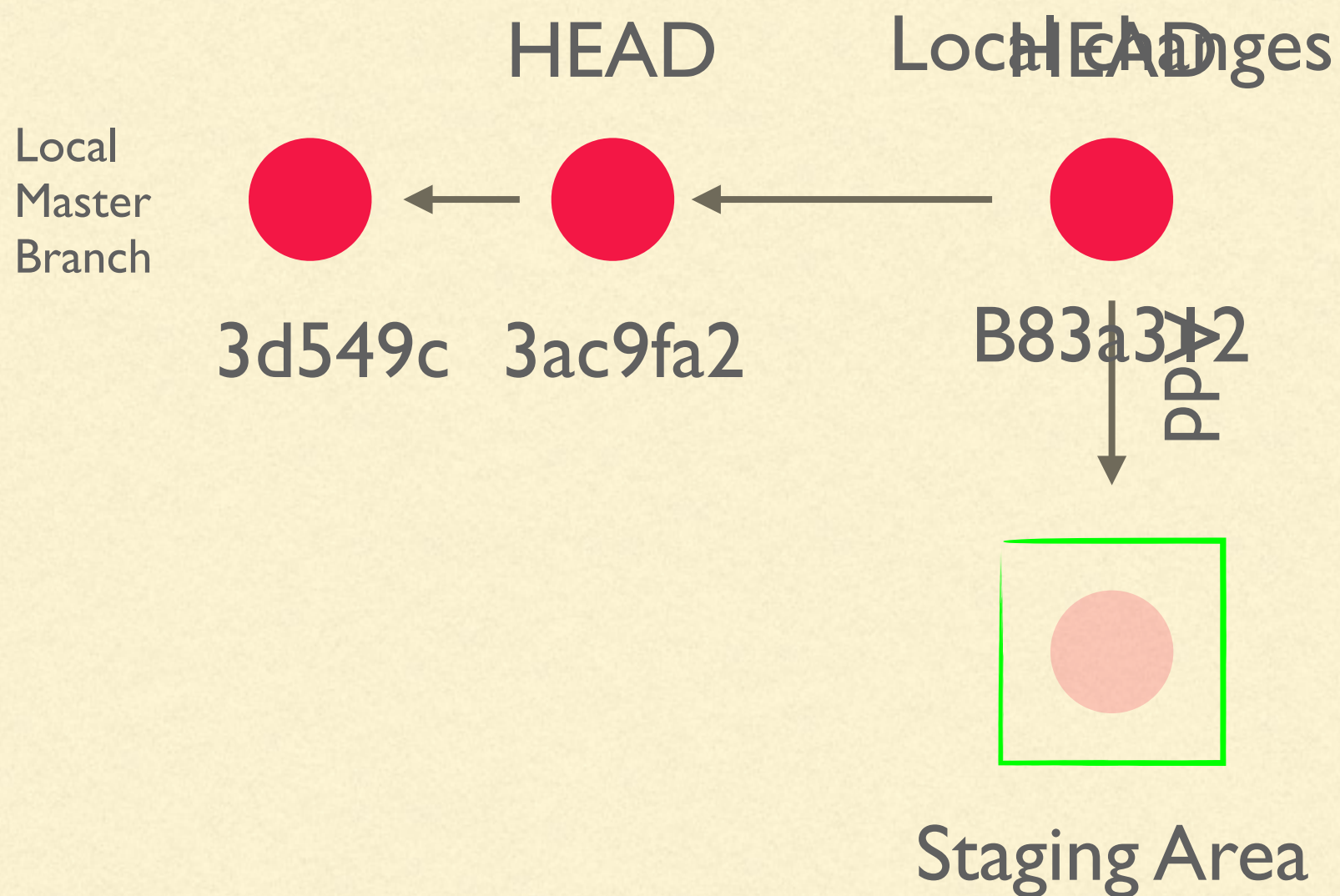
# \$ git add ...

---

- \$ 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



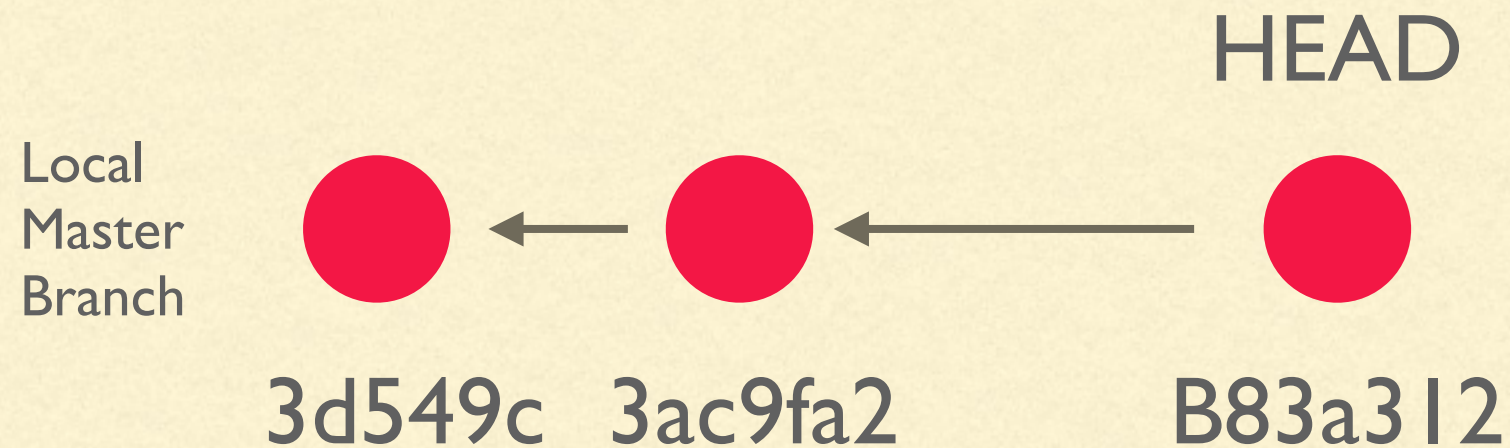
```
$ git commit -m "Add meaningful message"
```



---

# 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<sup>^</sup>, HEAD<sup>^^</sup>

---

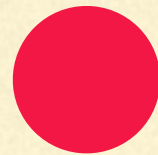


---

# 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>
```

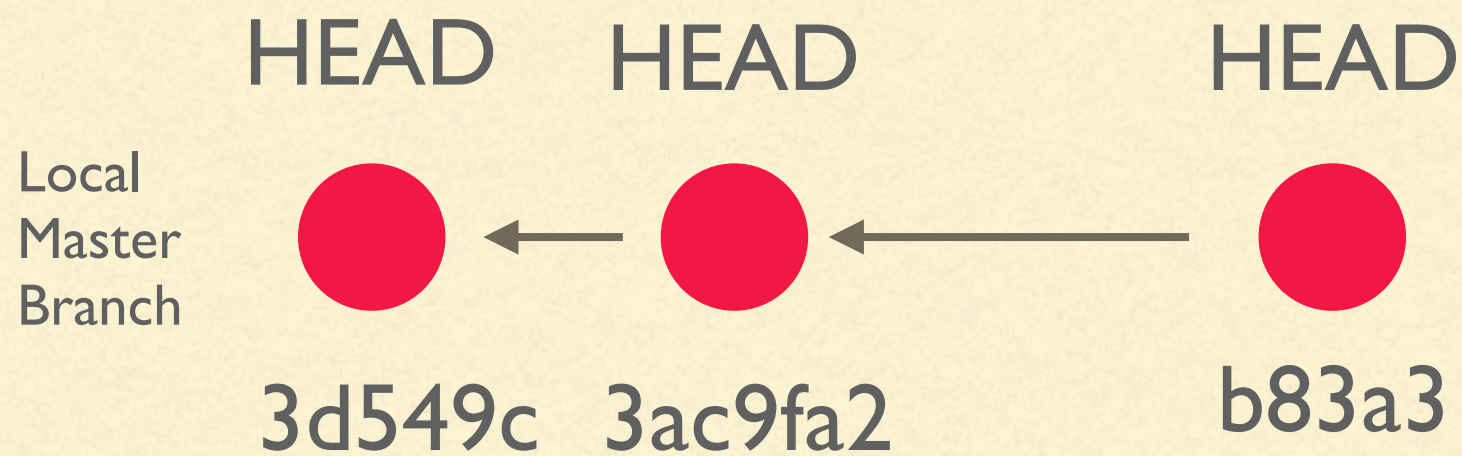
---



---

# 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

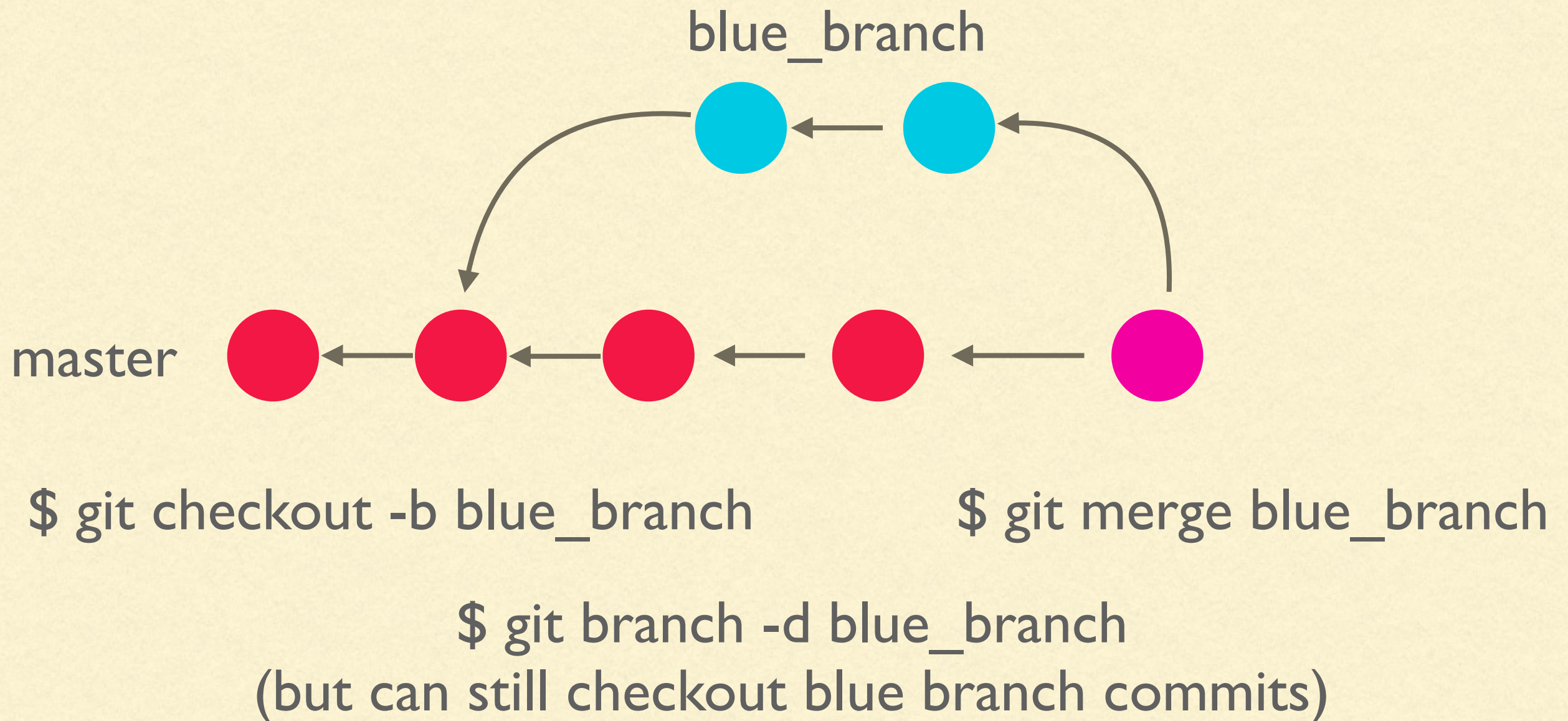
---



---

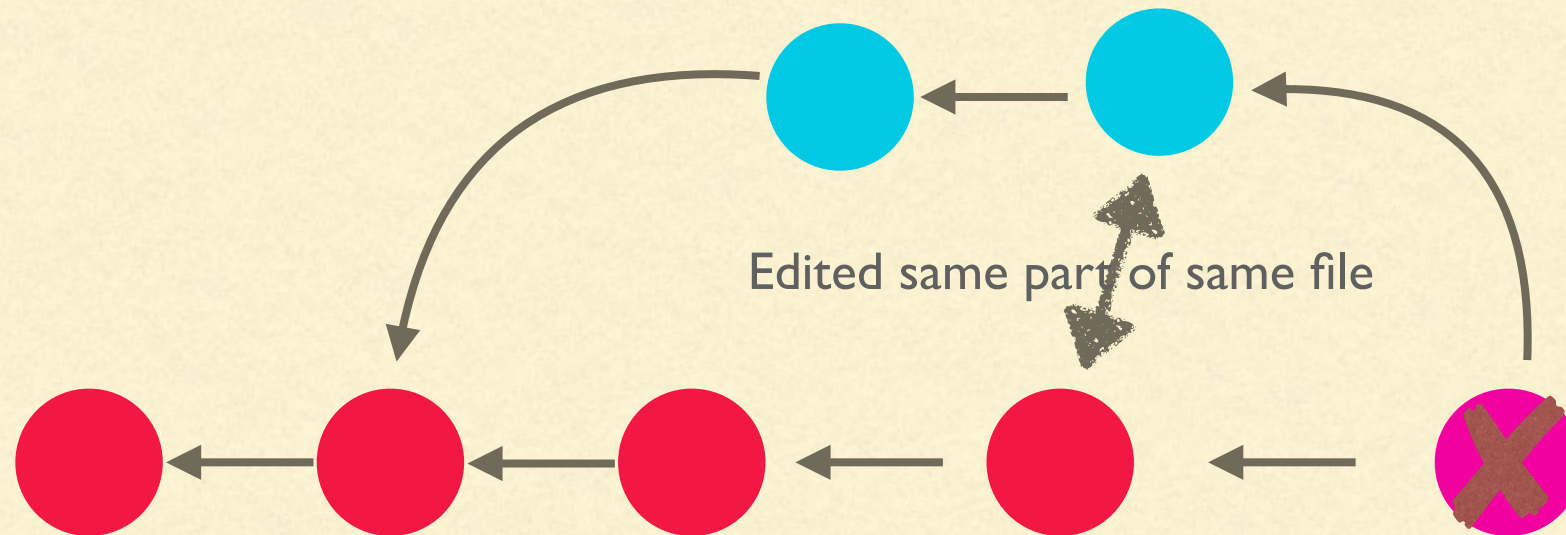
# Branches

---





# Branches - merge conflict





---

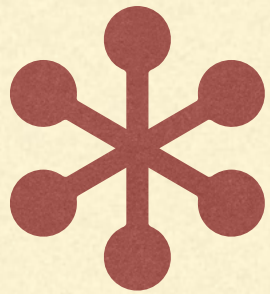
# Branches - merge conflict

```
<<<<<<< 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}

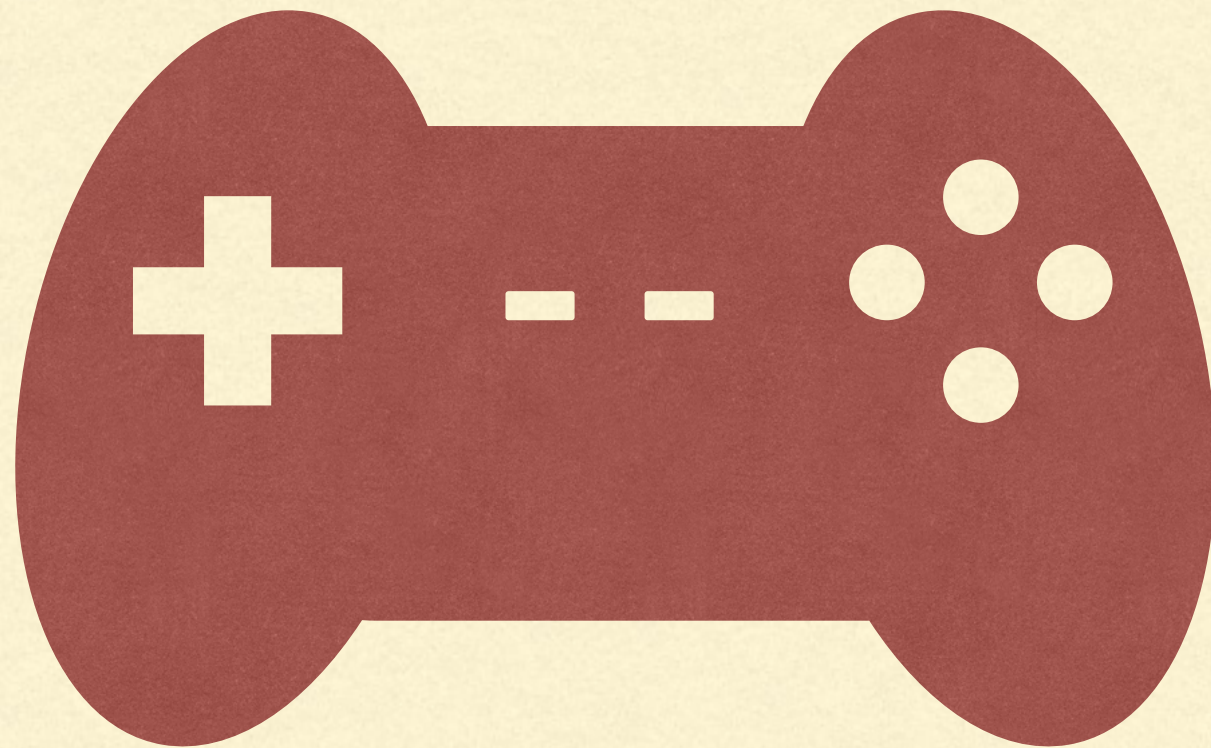
---



---

# 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

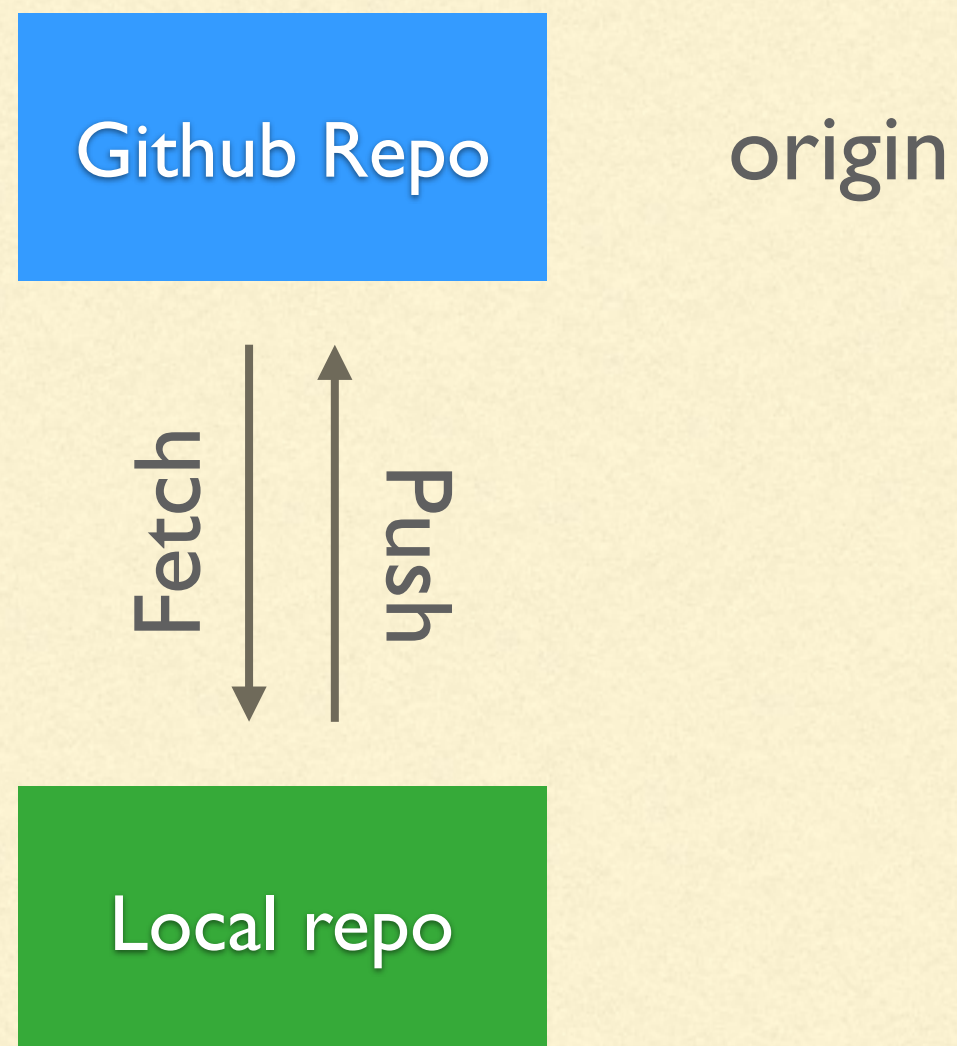
---



---

# Cloned repository structure

---





---

# Remote - origin

---

- `git remote -v`

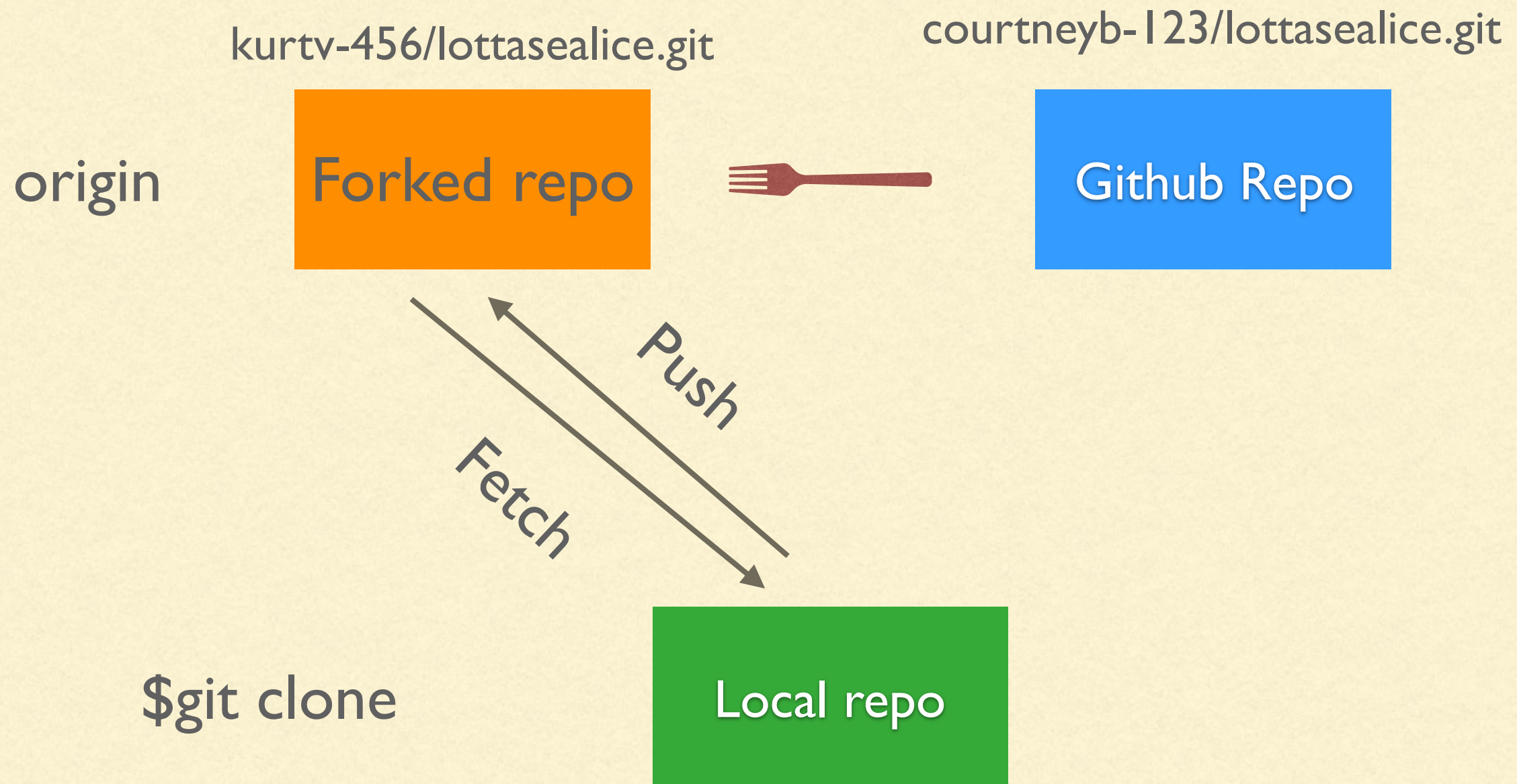
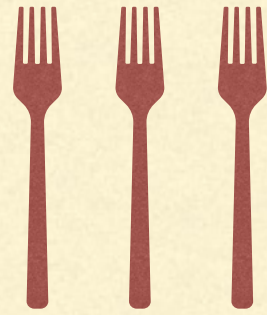
`$ origin https://github.com/courtneyb-l23/lottasealice.git (fetch)`

`$ origin https://github.com/courtneyb-l23/lottasealice.git (push)`

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



# Forks

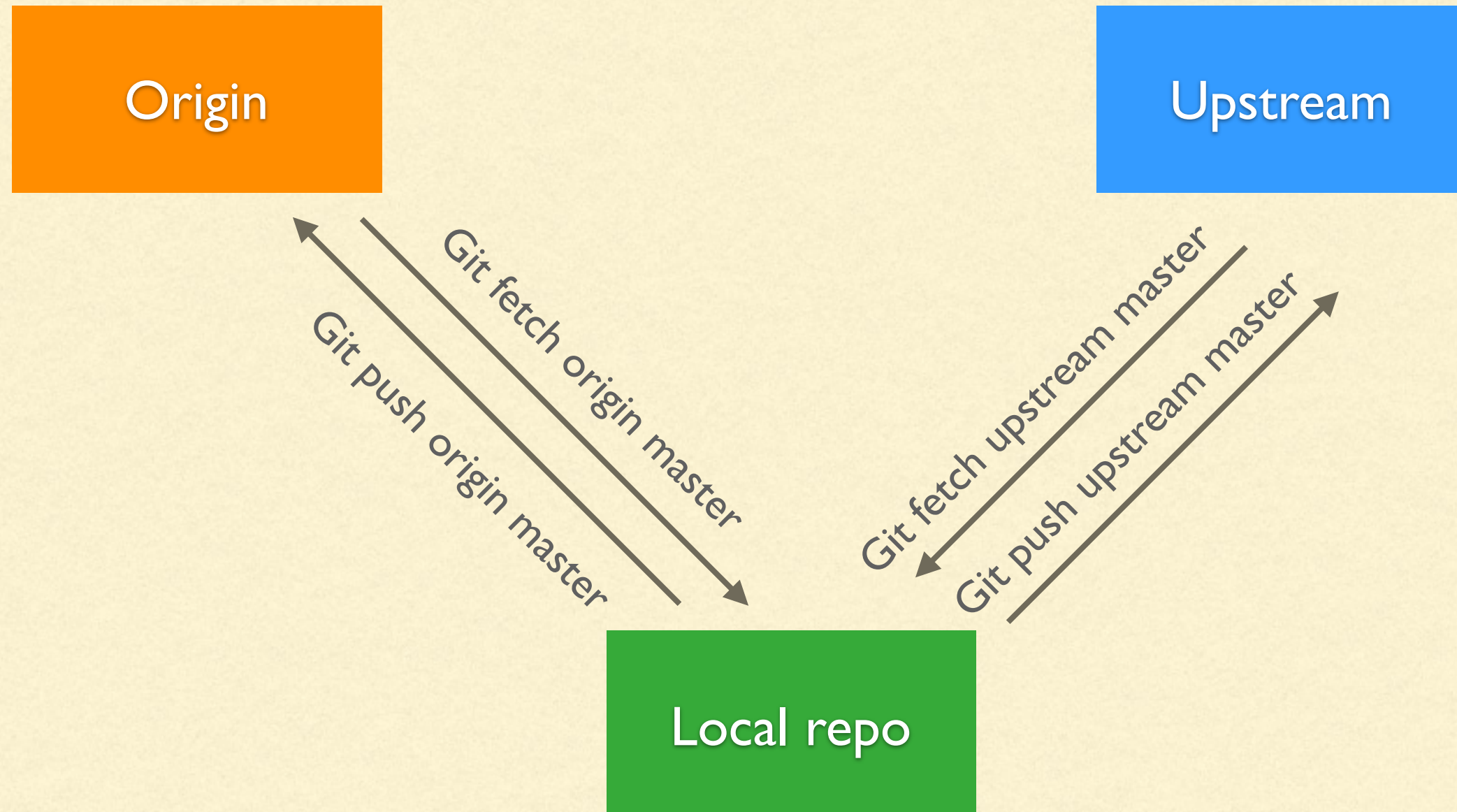




---

kurtv-456/lottasealice.git

courtneyb-123/lottasealice.git



```
$ 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

Git fetch <sup>remote</sup> upstream master  
branch

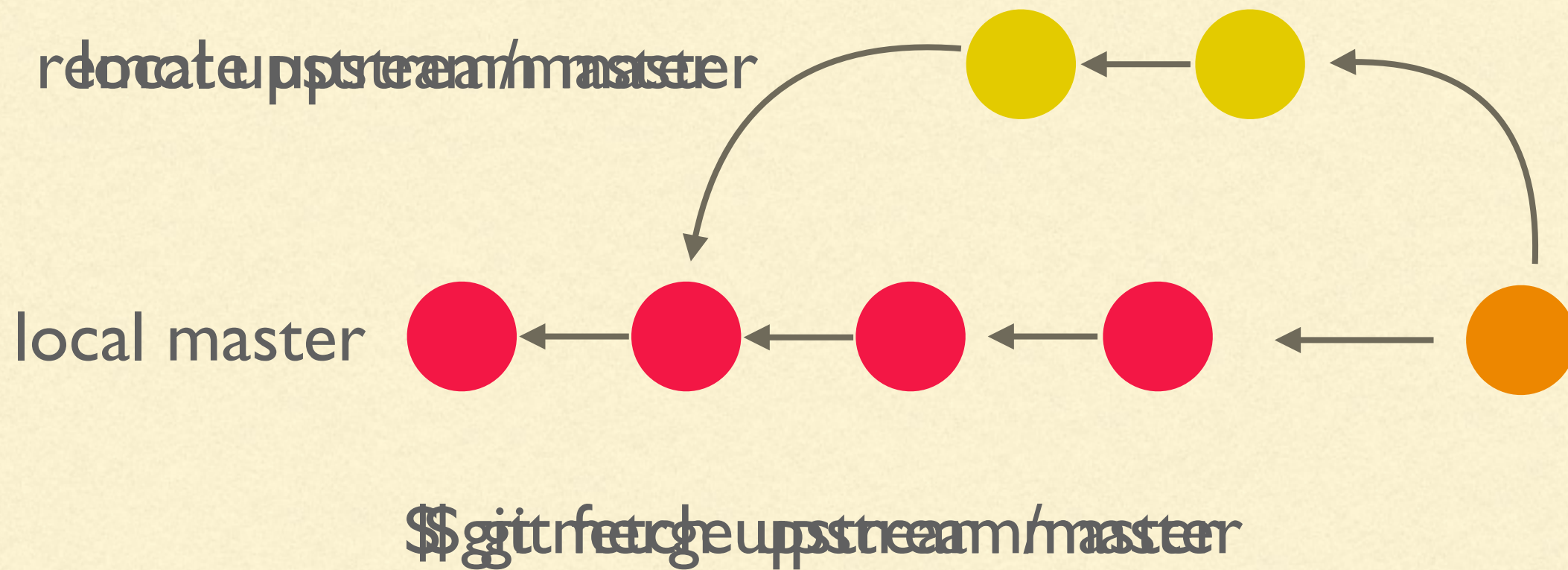
Why could it be batch to fetch without branch?

All changes across branches will be pulled

---



# Branches and forks





---

# 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
-