

# Version Control II

Bernease Herman<sup>1,2</sup>, David Beck<sup>1,2,3</sup>

<sup>1</sup>eScience Institute

<sup>2</sup>Computer Science

<sup>2</sup>Chemical Engineering

October 15, 2020



# **Review from first version control lecture**



# Time to remember your GitHub logins



# INTRODUCTION TO GIT

#(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > .gitignore

# INTRODUCTION TO GIT

#(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > .gitignore

## 1. Make changes



(use your preferred editor and tools.)

# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > .gitignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create Snapshot

- > git commit
- > git commit -m "[msg]"



# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create Snapshot

- > git commit
- > git commit -m "[msg]"





# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create Snapshot

- > git commit
- > git commit -m "[msg]"



## 4. Explore

- > git status
- > git log [options]
- > git show [sha1]

(Repeat 1-4 as desired.)

# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create Snapshot

- > git commit
- > git commit -m "[msg]"



auto



## 4. Explore

- > git status
- > git log [options]
- > git show [sha1]

(Repeat 1-4 as desired.)

# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create Snapshot

- > git commit
- > git commit -m "[msg]"



## 4. Explore

- > git status
- > git log [options]
- > git show [sha1]

## 5. Add remote

- > git remote add [name][url]
- > git remote -v



(Repeat 1-4 as desired.)

# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create Snapshot

- > git commit
- > git commit -m "[msg]"



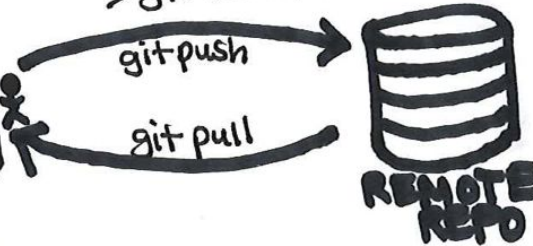
auto



LOCAL  
REPO

## 5. Add remote

- > git remote add [name][url]
- > git remote -v



## 4. Explore

- > git status
- > git log [options]
- > git show [sha1]

(Repeat 1-4 as desired.)



# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create snapshot

- > git commit
- > git commit -m "[msg]"



auto



## 4. Explore

- > git status
- > git log [options]
- > git show [sha1]

## 5. Add remote

- > git remote add [name][url]
- > git remote -v

git push



## 6. Pull from remote

- > git fetch [remote][branch]
- > git pull [remote][branch]

(Repeat 1-4 as desired.)

# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create snapshot

- > git commit
- > git commit -m "[msg]"



auto



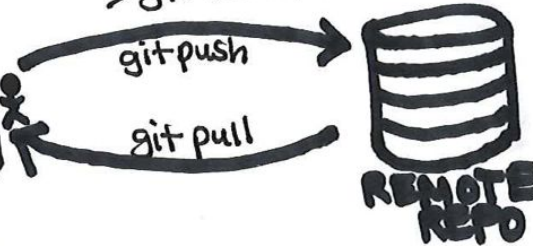
LOCAL REPO

## 4. Explore

- > git status
- > git log [options]
- > git show [sha1]

## 5. Add remote

- > git remote add [name][url]
- > git remote -v



## 6. Pull from remote

- > git fetch [remote][branch]
- > git pull [remote][branch]

## 7. Push to remote

- > git push [remote][branch]

(Repeat 1-4 as desired.)

**Done with review, on to new material**



# INTRODUCTION TO GIT

\*(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create snapshot

- > git commit
- > git commit -m "[msg]"

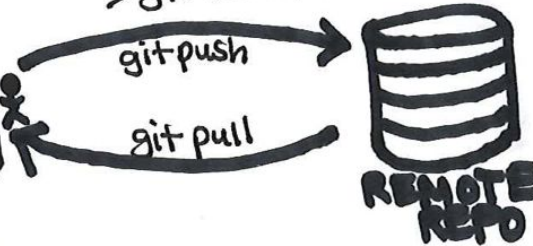


## 4. Explore

- > git status
- > git log [options]
- > git show [sha1]

## 5. Add remote

- > git remote add [name][url]
- > git remote -v



## 6. Pull from remote

- > git fetch [remote][branch]
- > git pull [remote][branch]

## 7. Push to remote

- > git push [remote][branch]

(Repeat 1-4 as desired.)



# INTRODUCTION TO GIT

#(and some GitHub)

## 0. Set up

- > git config [options]
- > git init
- > git ignore

## 1. Make changes



(use your preferred editor and tools.)

## 2. Stage changed files

- > git add
- > git add -A
- > git rm [path]



## 3. Create Snapshot

- > git commit
- > git commit -m "[msg]"



## 4. Explore

- > git status
- > git log [options]
- > git show [sha1]

(Repeat 1-4 as desired.)

## 8. Undoing changes

- > git reset [options]
- > git revert [sha1]

## 9. Rewriting history

(Not to be used on public commits!)

- > git commit --amend
- > git rebase [-i]
- > git reflog

TIP: pull before commit to minimize conflicts!

## 10. Climbing the Git tree



- > git checkout

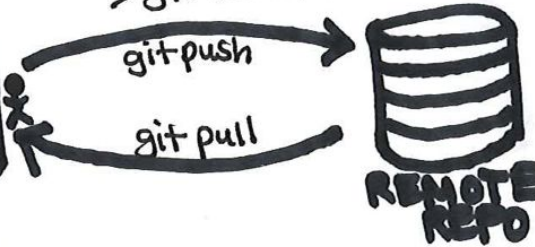
Detached HEAD state!

## BONUS: Conflicts

- > git merge
- > git rebase

## 5. Add remote

- > git remote add [name][url]
- > git remote -v



## 6. Pull from remote

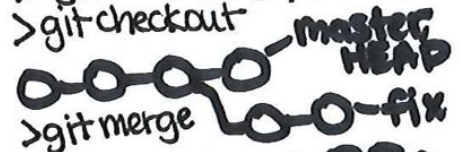
- > git fetch [remote][branch]
- > git pull [remote][branch]

## 7. Push to remote

- > git push [remote][branch]

## 11. Branches

- > git branch [options]
- > git checkout



## 12. Forks and PRs



## 13. Workflows and Tags and More

- > git tag [options]

Bernease Herman 10/4/18

# A single commit



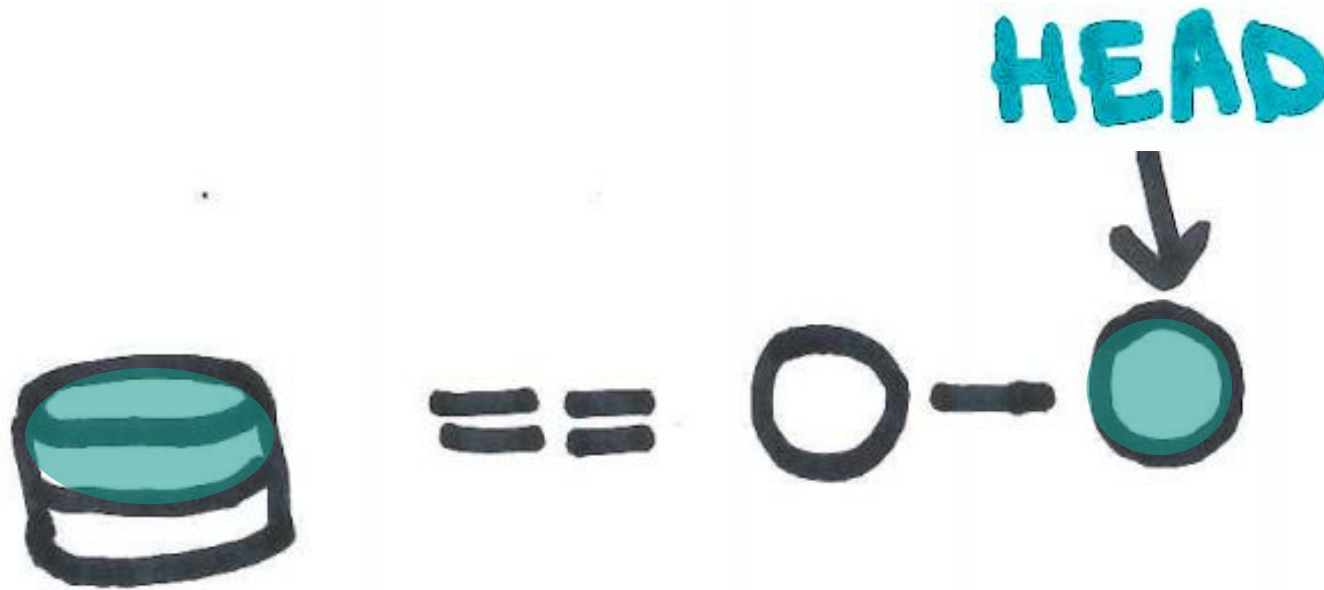
# In tree representation



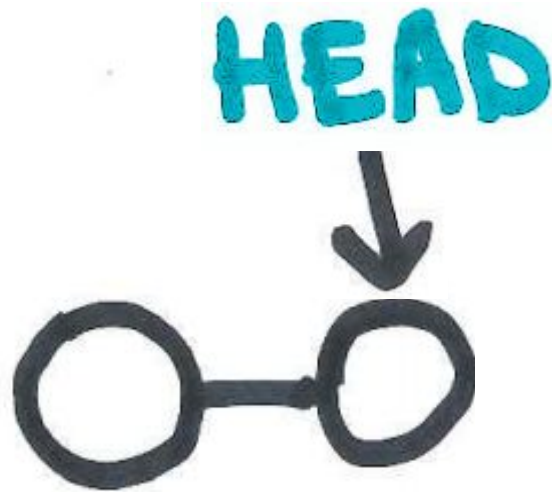
# Multiple commits represented



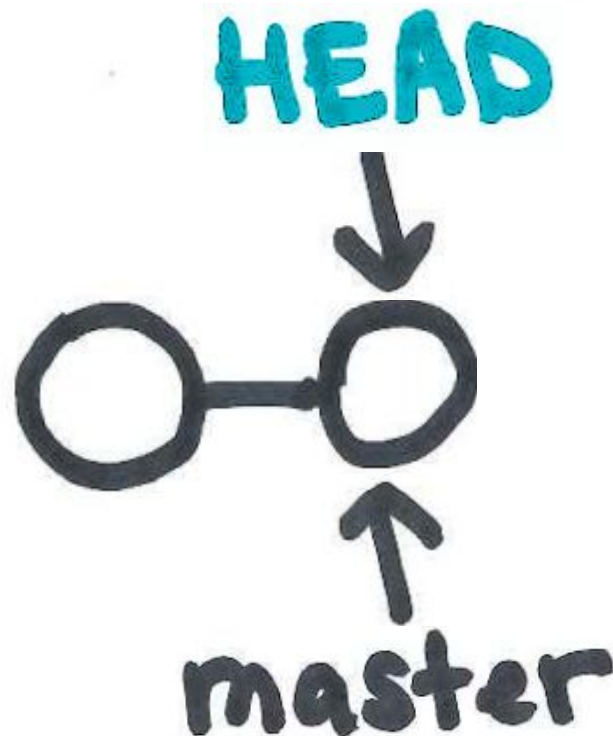
# Your working directory and files



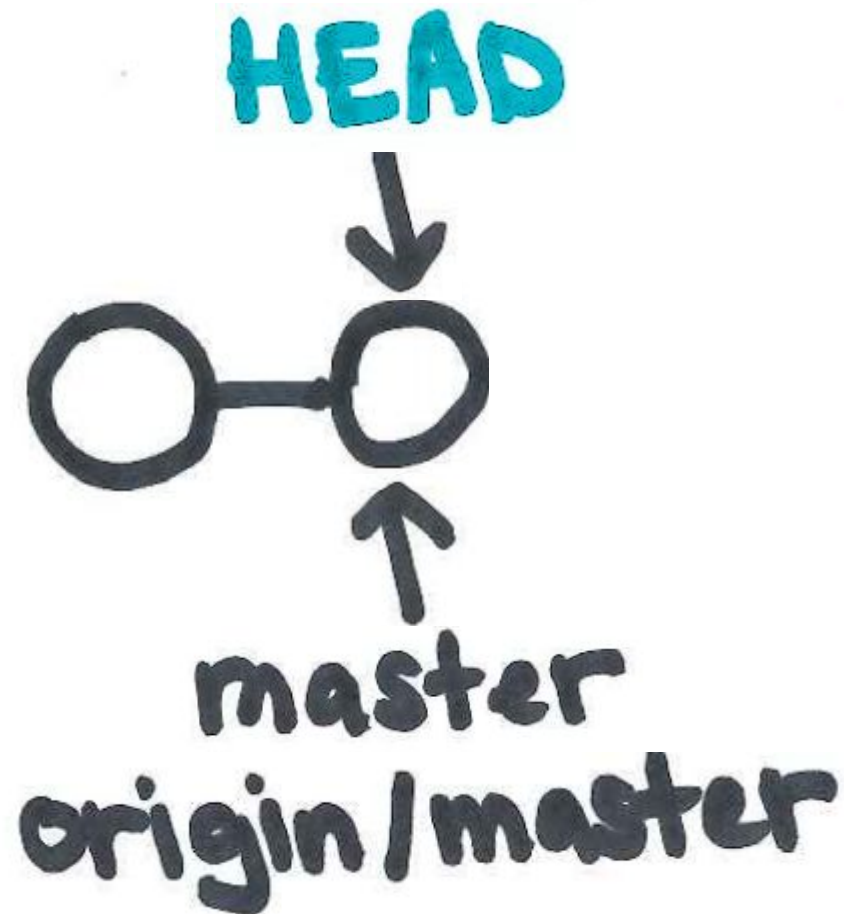
# HEAD pointer on our tree



**Think of branches as a pointer, as well**

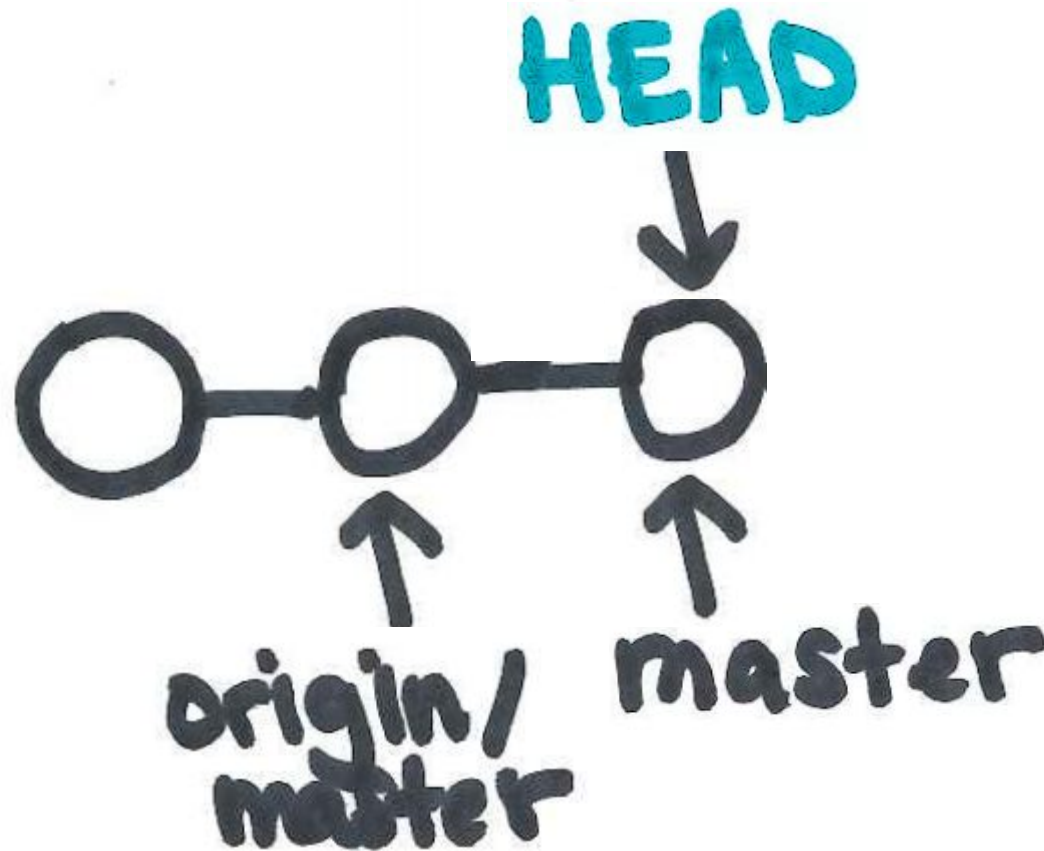


# Remote branches are included





# Local commit, before pushing to remote



# Editing and Deleting Commits

**git amend:** Allows you to add new changes to the last commit. More options with rebase.

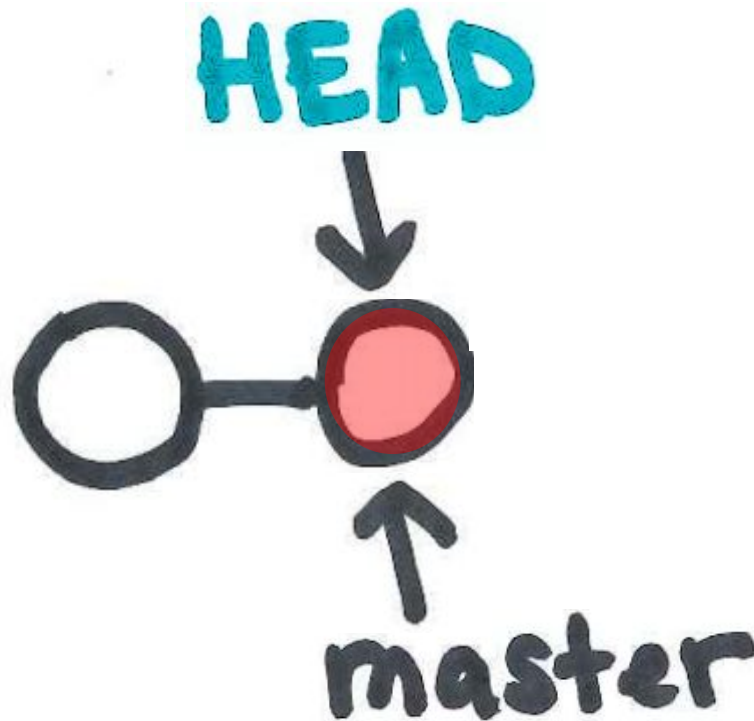
**git rebase [-i]:** Allows you to rename, squash, delete commits.

**git reset:** Removes a commit, staged changes, and working directory changes to delete history.

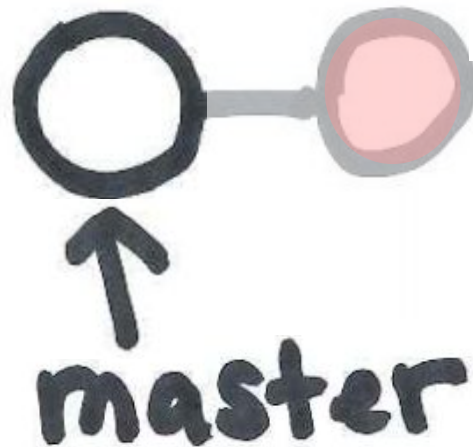
**git revert:** Creates an additional commit that reverses changes for specified commits. Good for public repos.



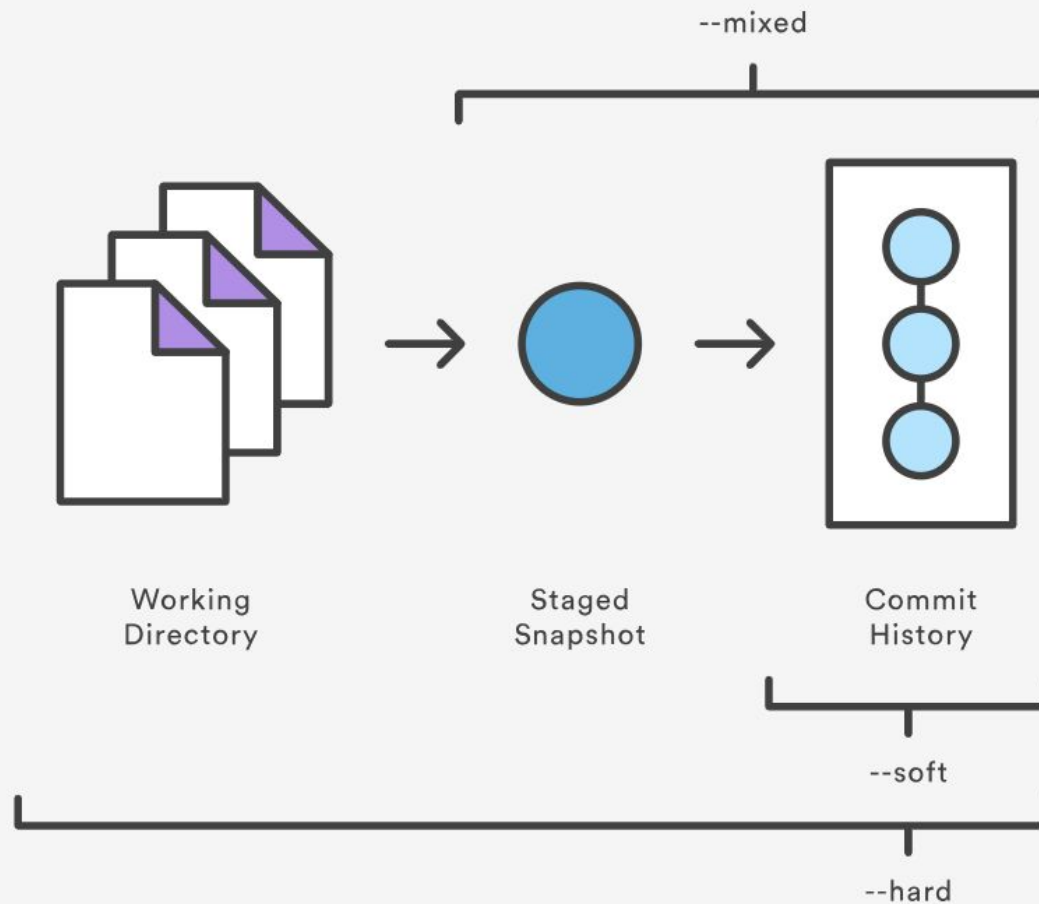
**You've committed an unwanted change  
(hiding origin/master for simplicity)**



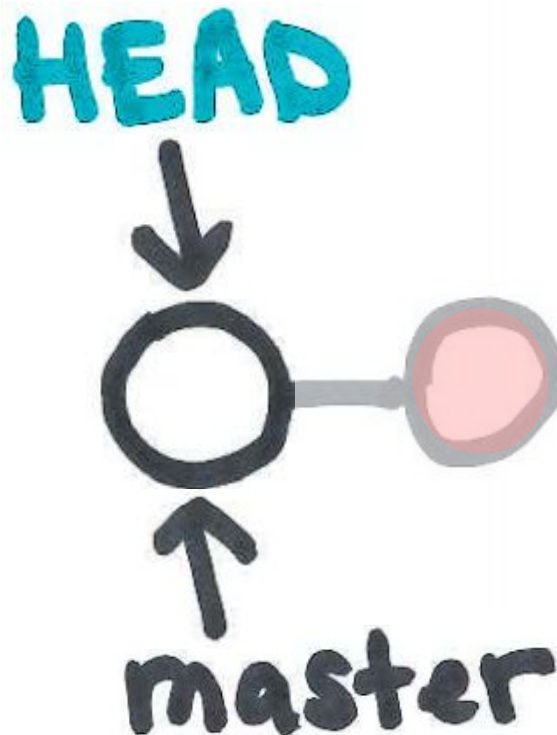
**If not public, reset your commit**



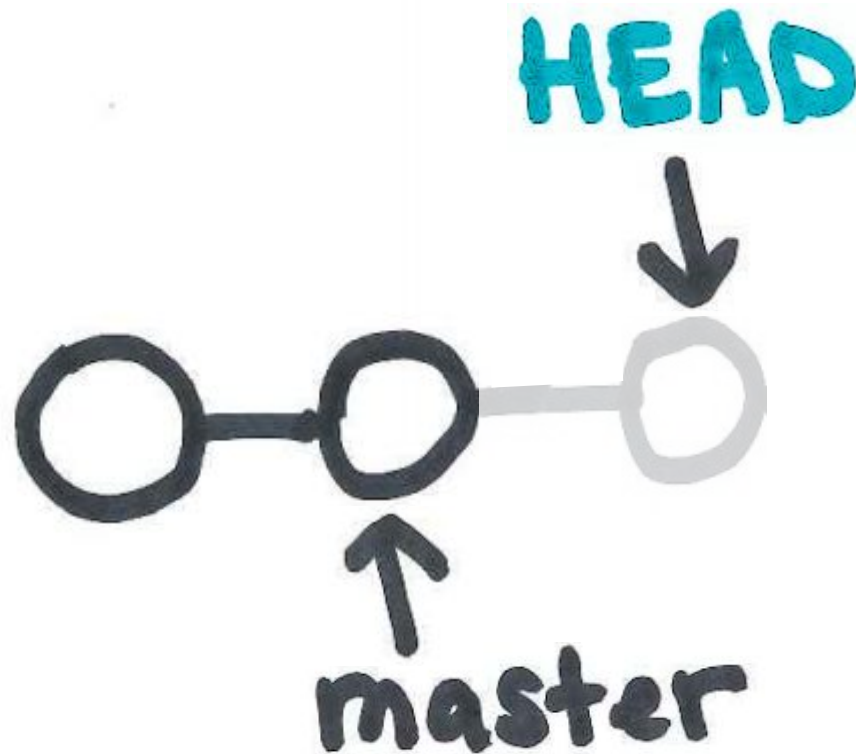
# Differing levels of reset



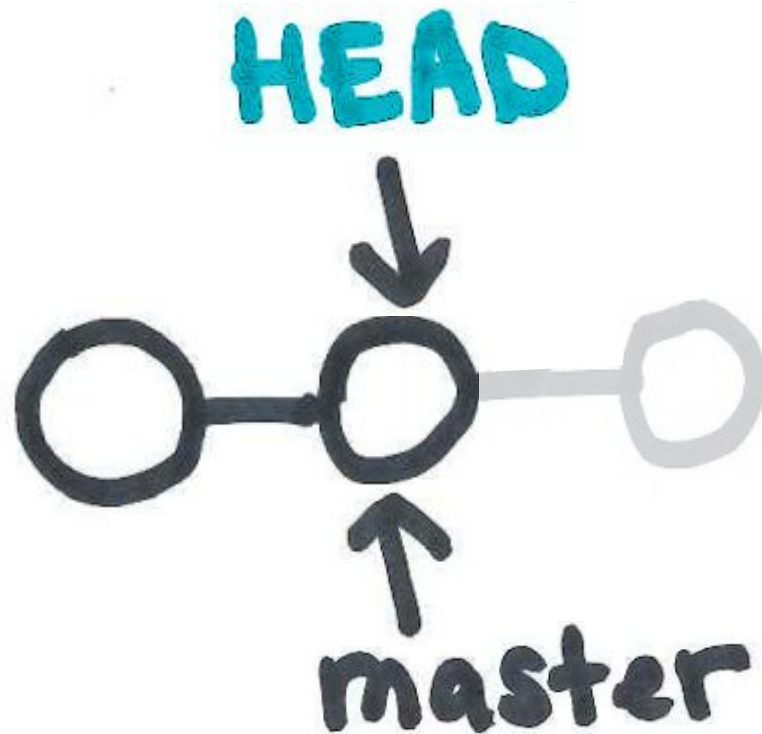
**Directory is unchanged for `git reset --hard` and `--mixed`, but not `--soft`.**



`git reset --soft/--mixed`

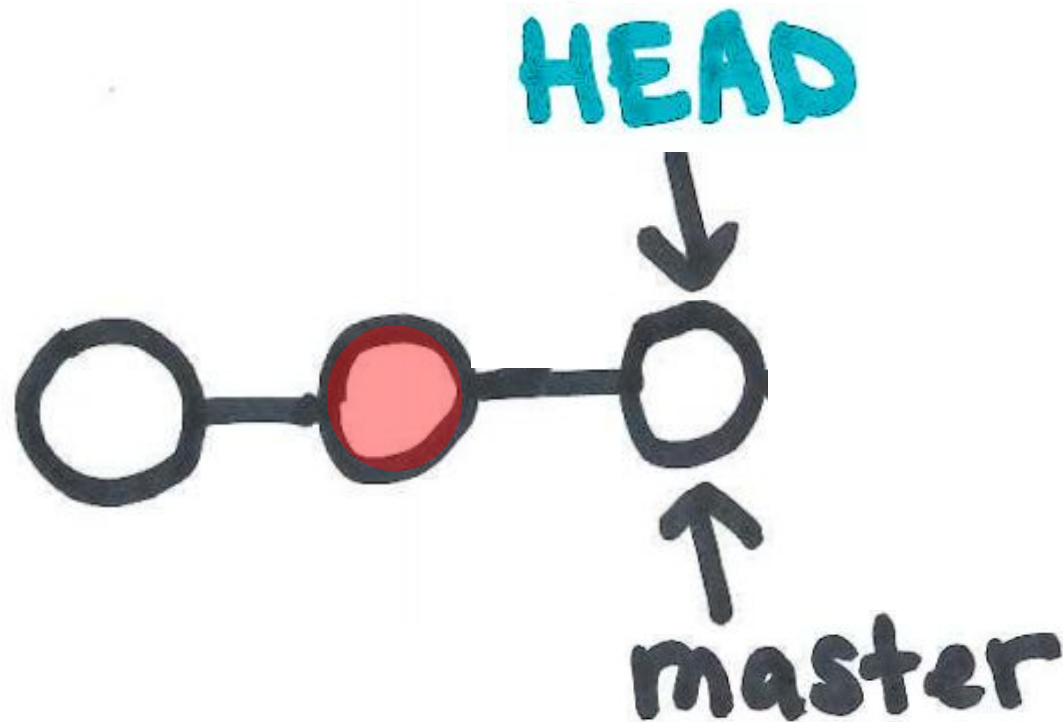


`git reset --hard`

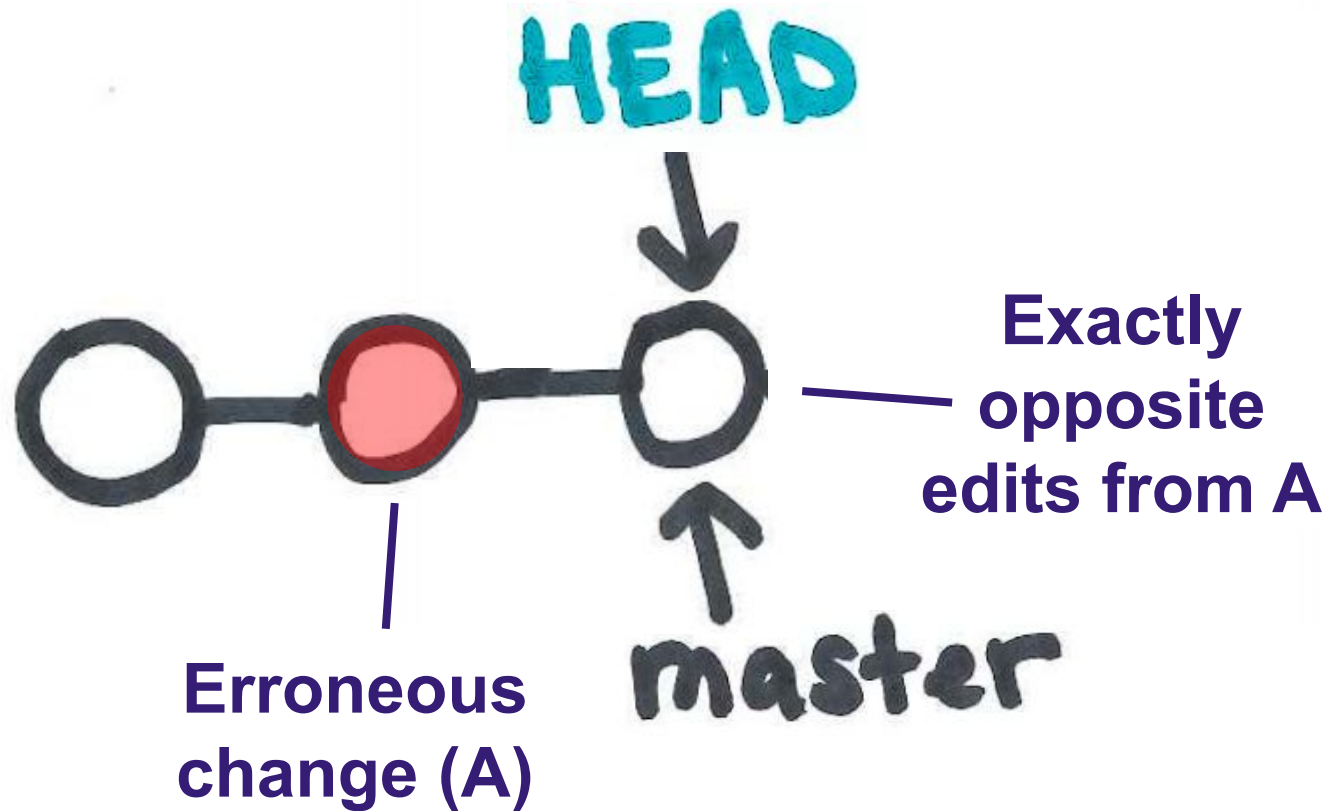




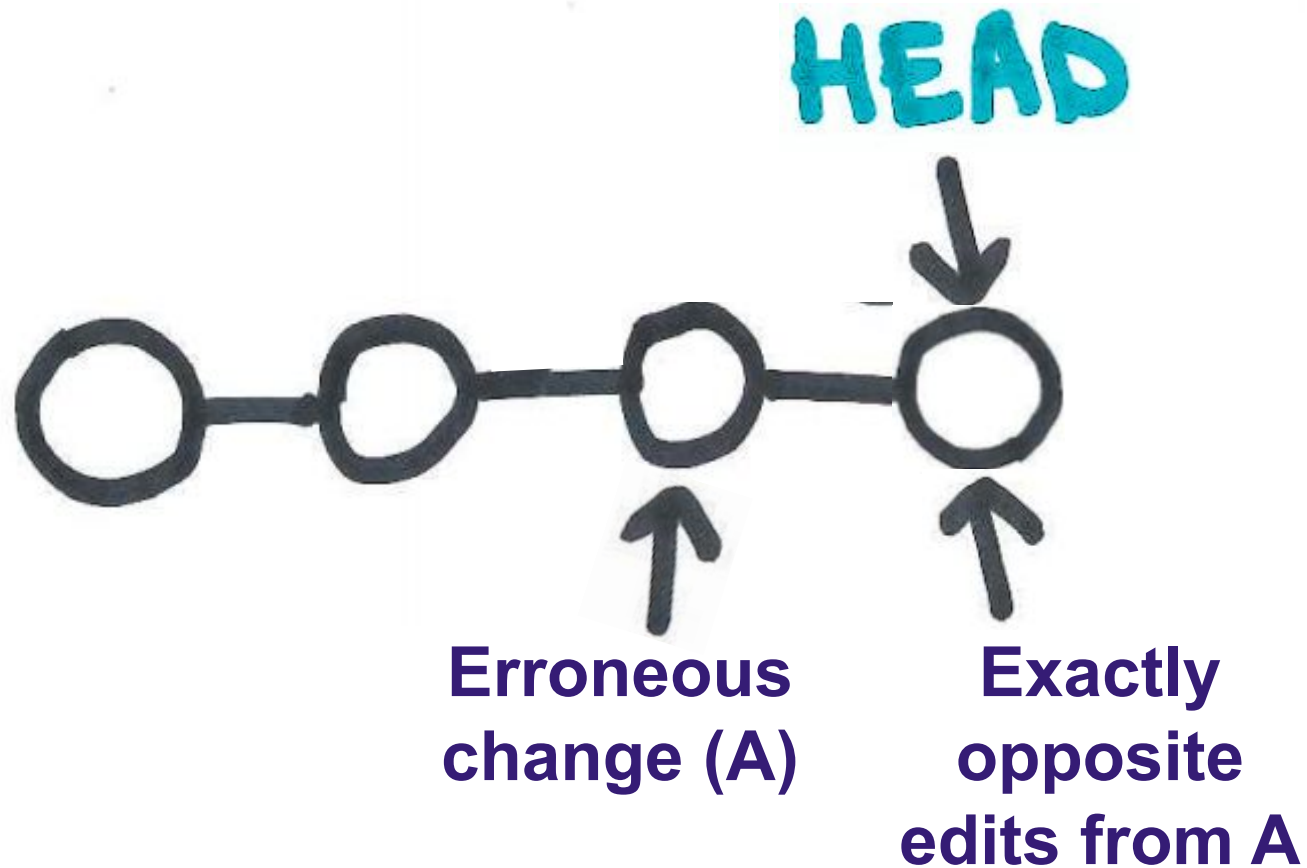
**If public, use git revert to add a new commit that fixes the issue.**



If public, use git revert to add a new commit that fixes the issue.



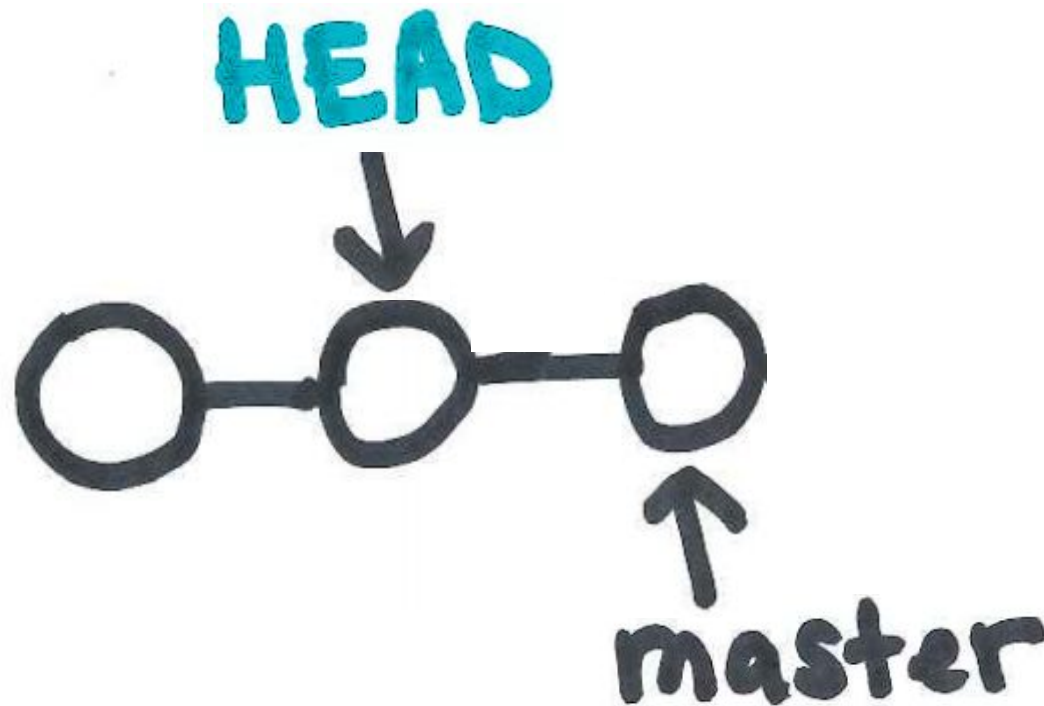
# Reverting a change when public



# A review of commands to fix changes

Command	Scope	Common use cases
git reset	Commit-level	Discard commits in a private branch or throw away uncommitted changes
git reset	File-level	Unstage a file
git checkout	Commit-level	Switch between branches or inspect old snapshots
git checkout	File-level	Discard changes in the working directory
git revert	Commit-level	Undo commits in a public branch
git revert	File-level	(N/A)

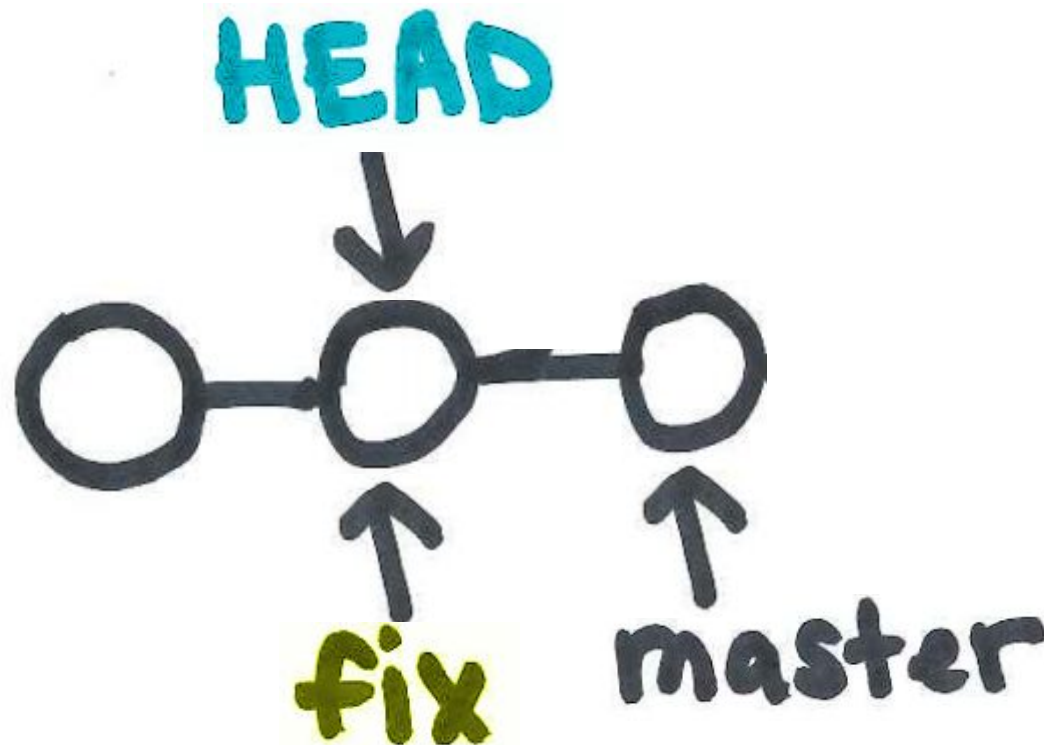
**Checkout an earlier commit  
(hiding origin/master for simplicity)**



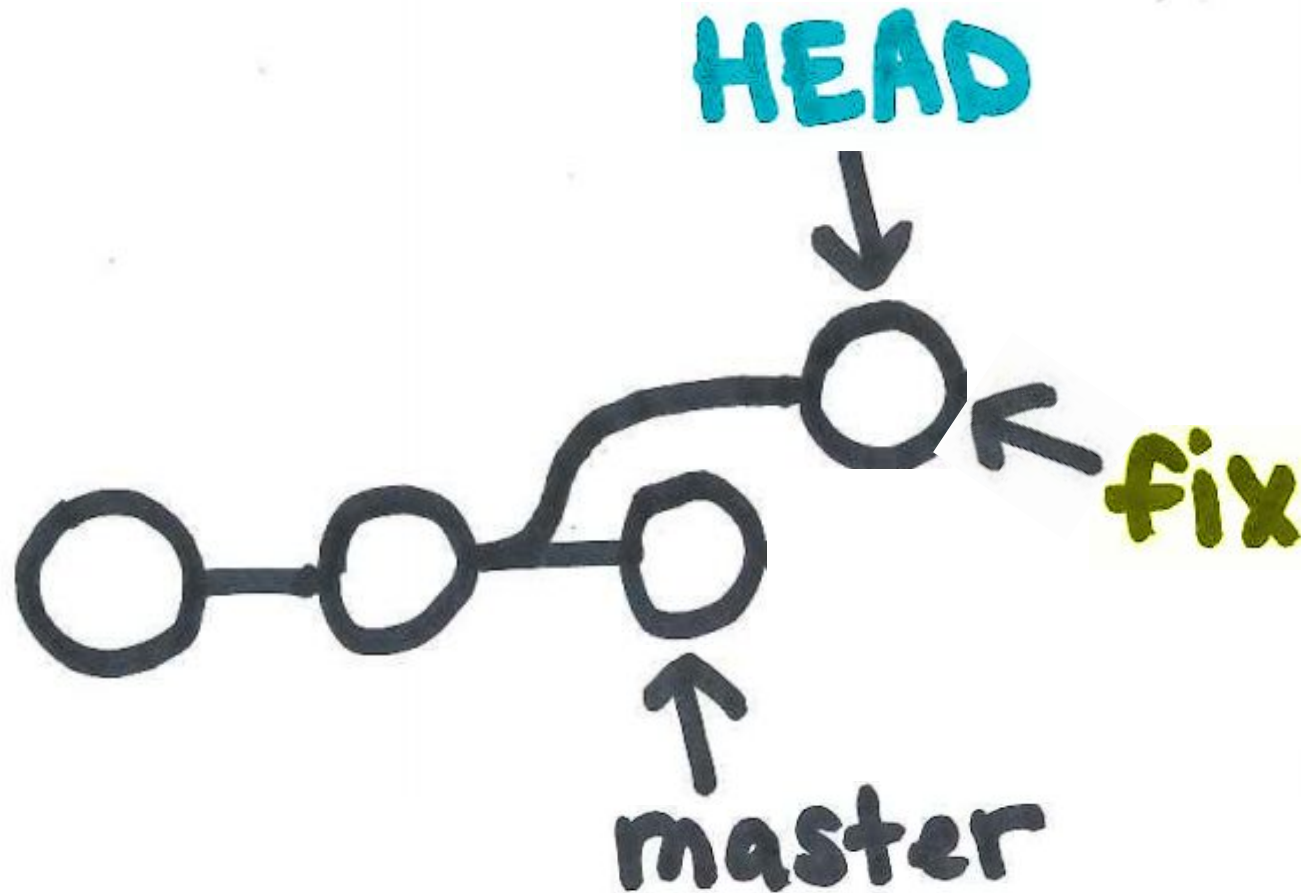
## Checking out a specific file

```
$ git checkout -- myfile.txt
```

# Creating a new branch

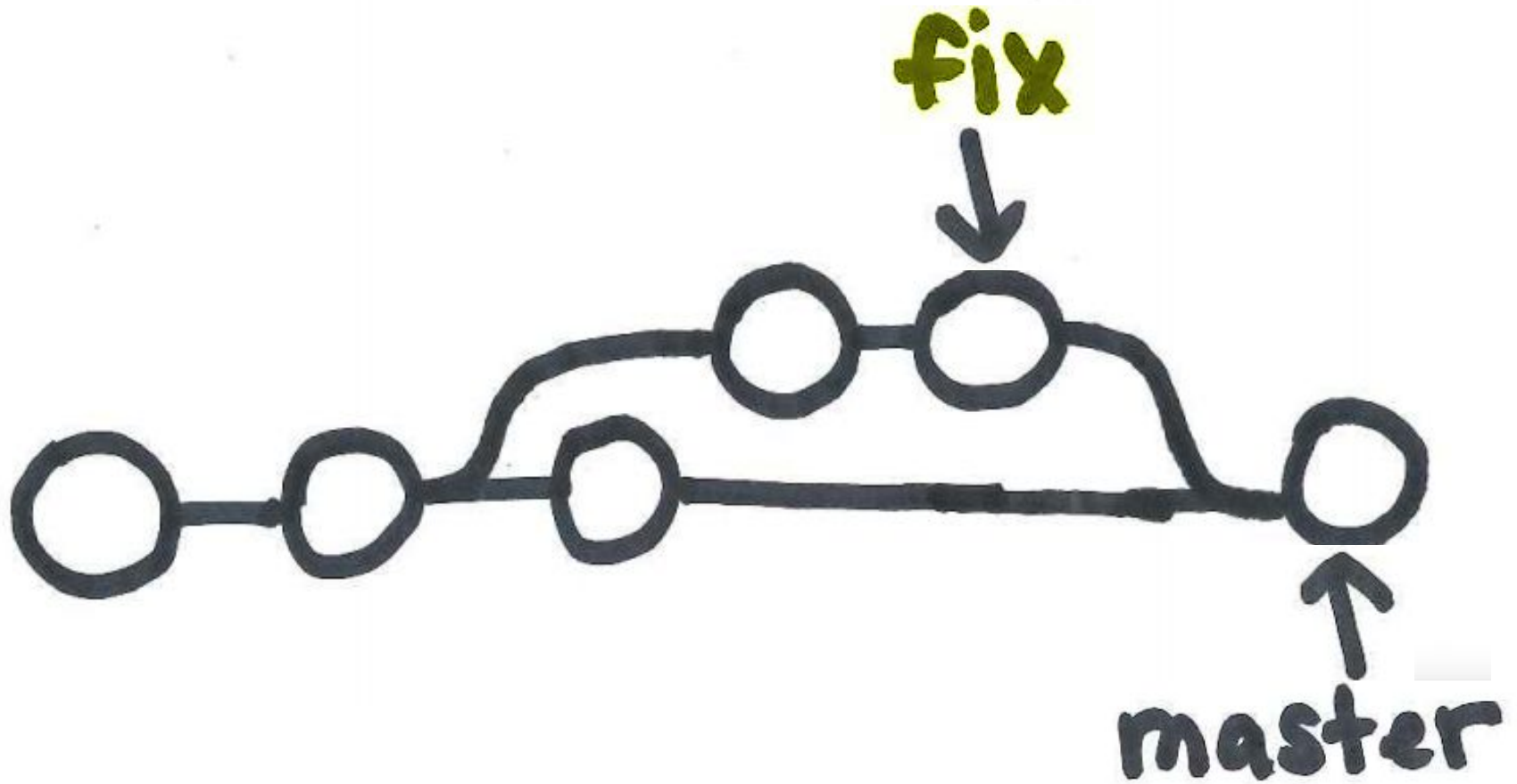


# Making changes along this branch





# Merging commits to another branch



# Exercise: Tracing the Git Tree

With a partner (or groups of 3), walk through how the following commands would change your git tree. Draw a diagram with the final tree that includes labels for HEAD, all local branches, and all remote branches (origin/\*).

Assume that all add/commit combinations has changes and creates a commit.

```
git init
```

```
git commit -a -m "First  
commit"
```

```
git commit -a -m "Second  
commit"
```

```
git remote add origin <url>  
(Assume remote has an empty repository.)
```

```
git push origin master
```

```
git checkout HEAD~1
```

```
git branch fix
```

```
git checkout fix
```

```
git commit -a -m "Third  
commit"
```

```
git push origin fix
```

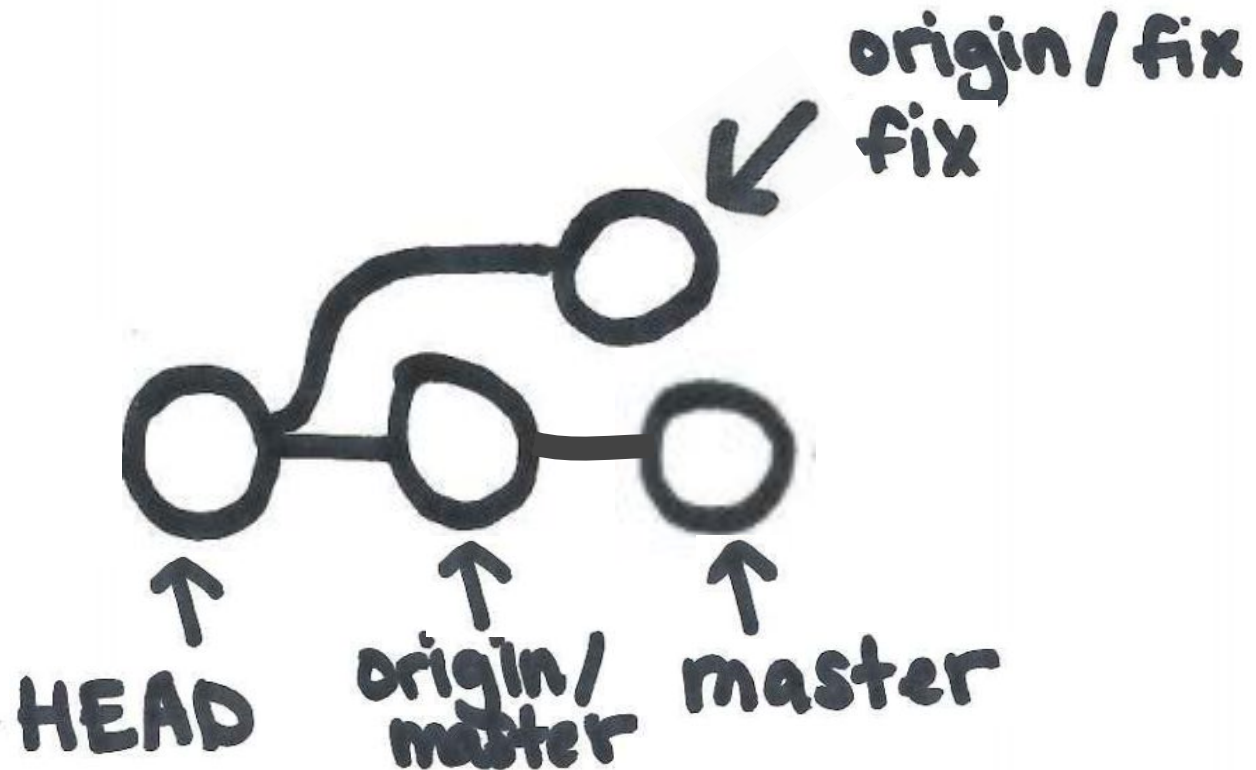
```
git checkout master
```

```
git commit -a -m "Fourth  
commit"
```

```
git checkout HEAD~2
```



## Exercise answer



# Collaboration workflows

Who should have permissions to push, pull, create repositories? Do we trust equally?

**Centralized workflow**

**Forking permissions workflow**

<https://www.atlassian.com/git/tutorials/comparing-workflows>



# Collaboration workflows

How complex are changes? Could they break the production system? How complex is the release schedule?

**Simple (forking) workflow**

**Feature branch workflow**

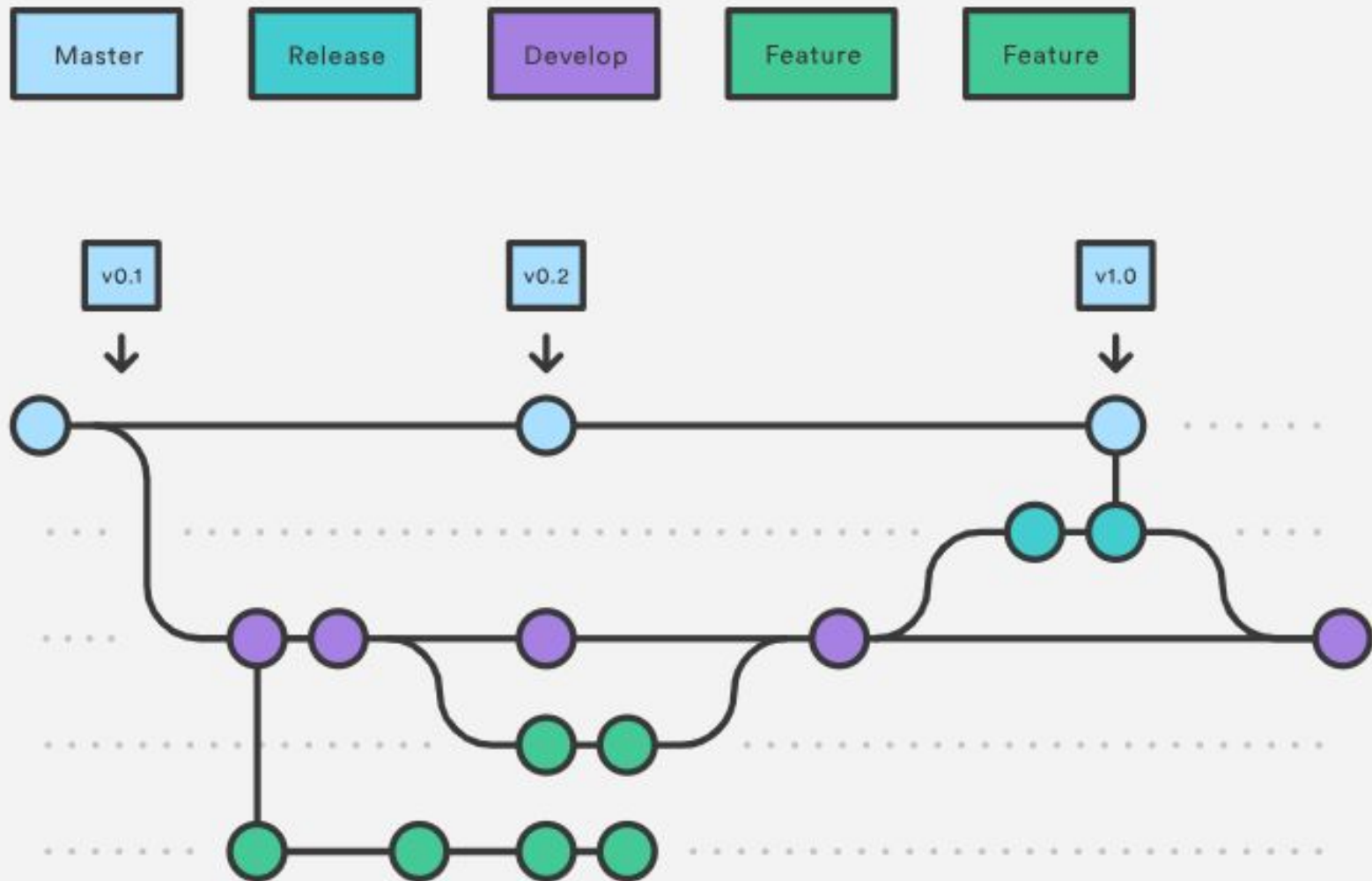
**Git flow workflow**

<https://www.atlassian.com/git/tutorials/comparing-workflows>



# Git flow workflow for larger projects

(image from Atlassian's online `git` tutorials)



# Questions?



origin /  
fix

origin / fix

origin /  
master

origin / master

origin / master

fix



master ↓

HEAD

