

The image features a futuristic, glowing blue digital interface set against a dark, metallic background. The interface consists of several glowing blue rectangular outlines forming a grid-like structure. In the center of this grid is a bright, jagged lightning bolt. Below the grid, the word "REAL" is written in a glowing blue, blocky font. The entire scene is set against a dark, textured background that resembles a circuit board or a futuristic control panel.

REAL

# GIT BASICS

LEVEL 1

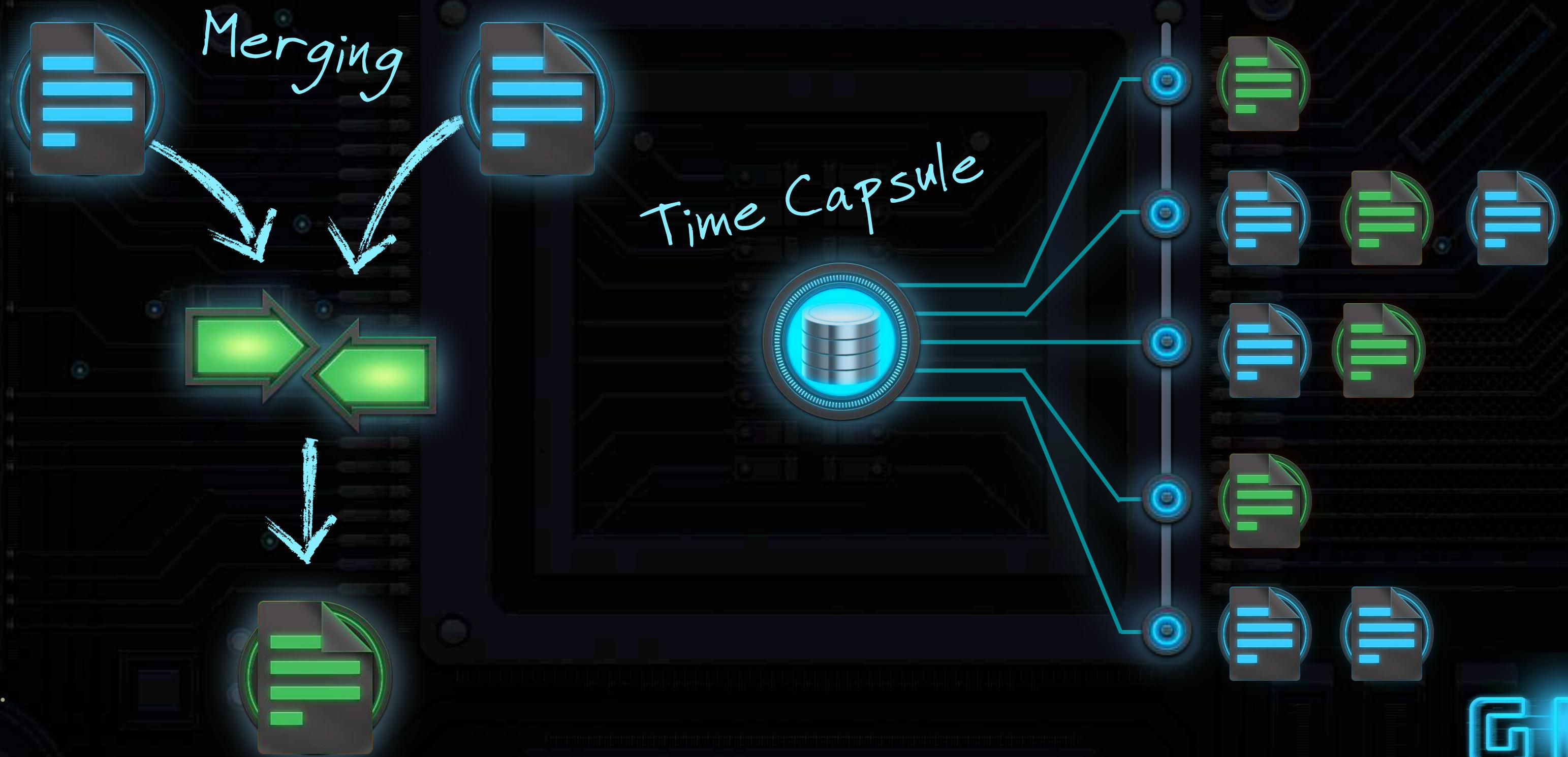
# SAME OLD STORY

Collaboration Issues!

LEVEL 1 – GIT BASICS

GIT  
REAL

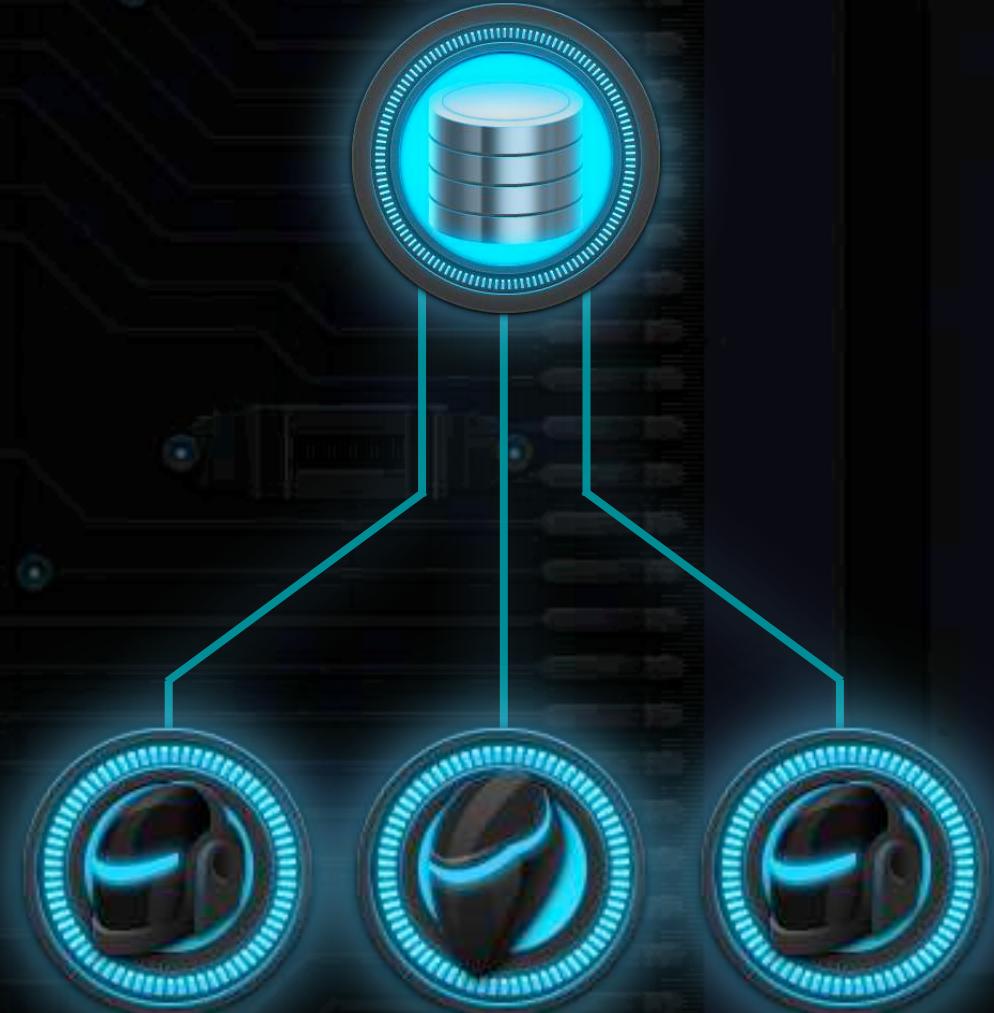
# VERSION CONTROL SYSTEMS



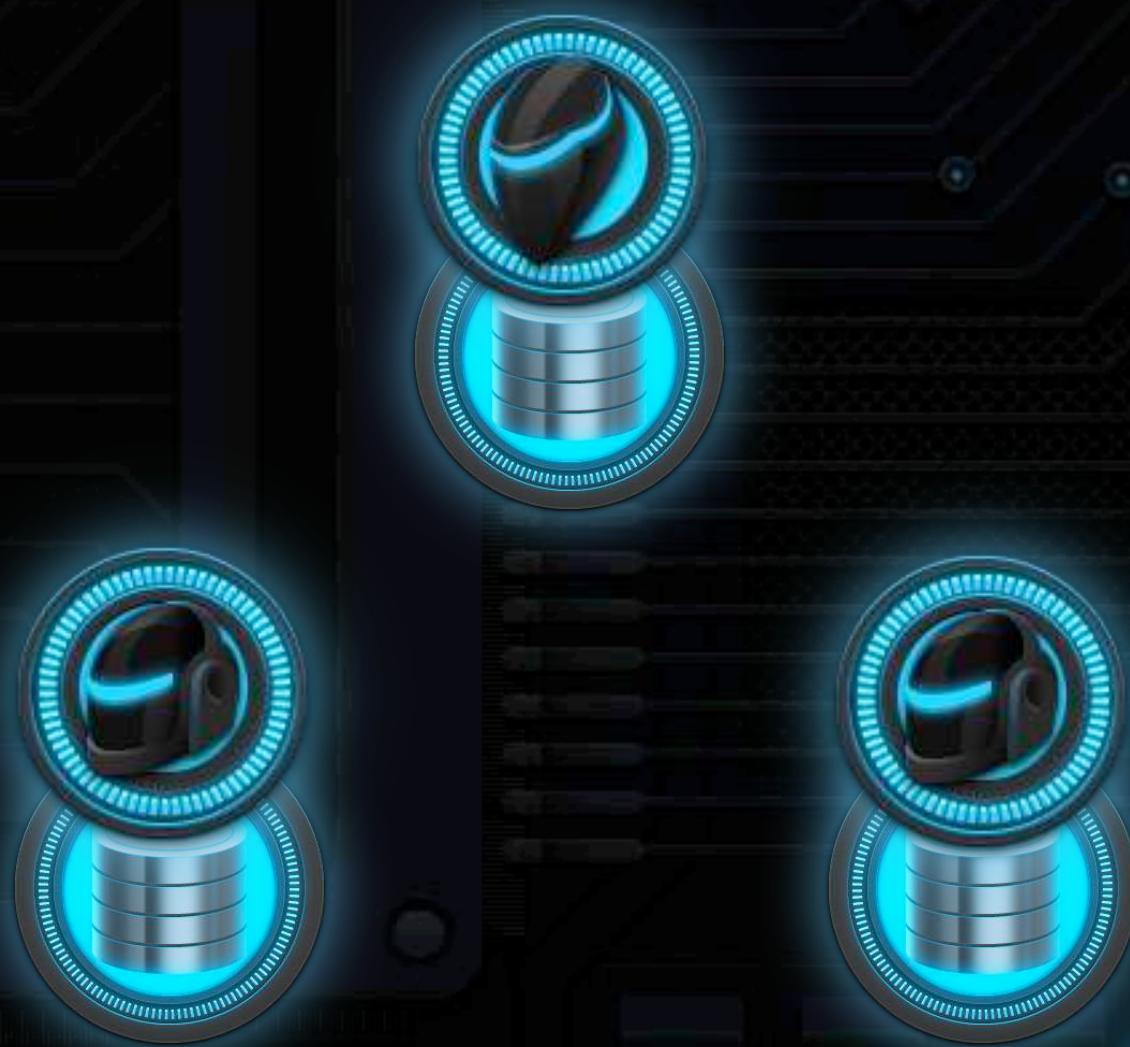
GIT  
REAL

# REPOSITORY LOCATIONS

Central Repository



Distributed Repository



**Git is a  
DISTRIBUTED VERSION CONTROL SYSTEM  
(DVCS)**

# ORIGINS

- Linux Kernel project.
- Meant to be distributed, fast and more natural.
- Capable of handling large projects.



LEVEL 1 – GIT BASICS



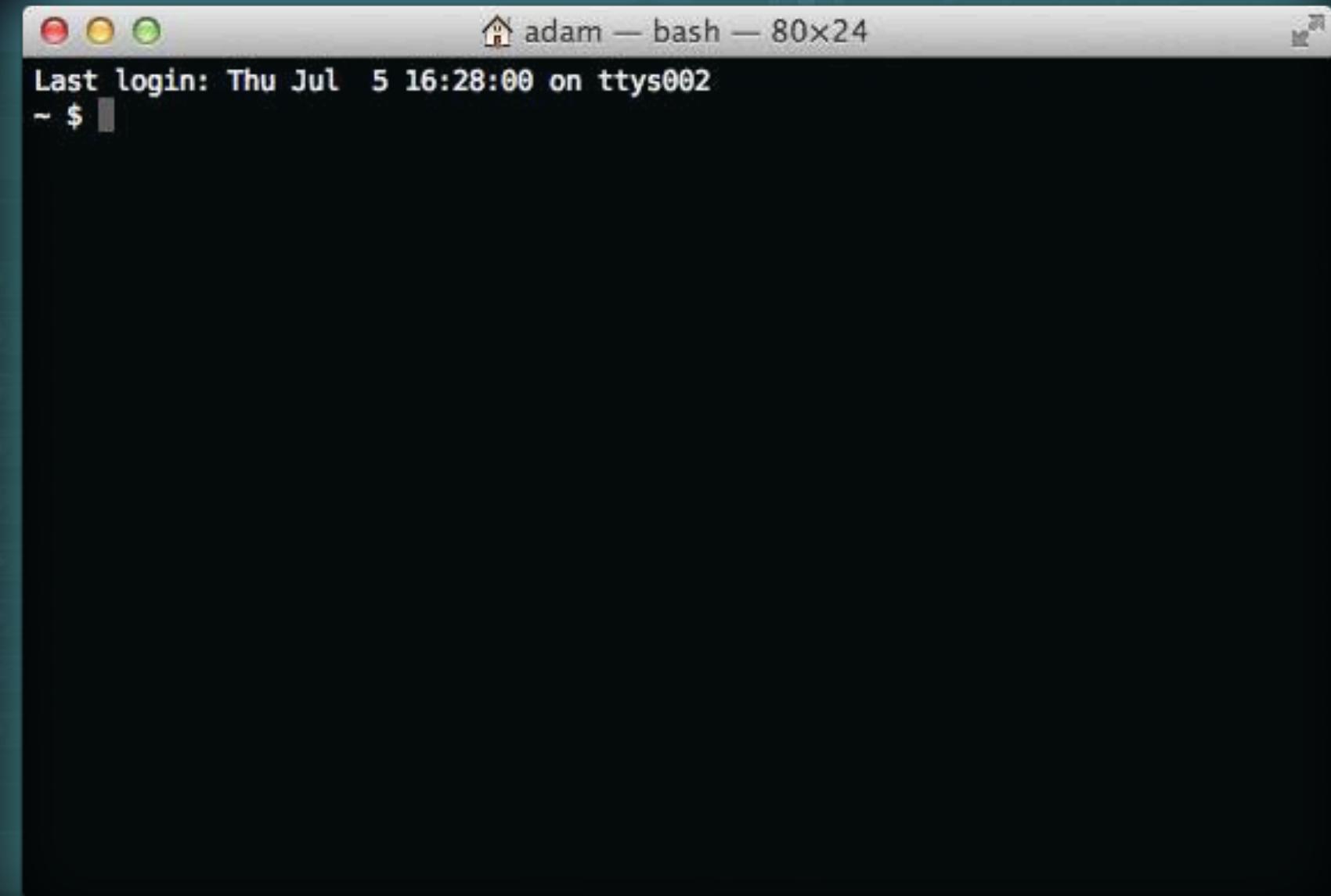
GIT  
REAL

# WORKING WITH GIT

- Command line interface
- Many GUI tools available
- Official Git Site —  
<http://git-scm.com/>

start here

LEVEL 1 – GIT BASICS



```
adam — bash — 80x24
Last login: Thu Jul  5 16:28:00 on ttys002
~ $
```

GIT  
REAL

# GIT HELP

```
$ git help  
usage: git [--version] [--exec-path[=<path>]] [--html-path]  
[-p|--paginate|--no-pager] [--no-replace-objects]  
[--bare] [--git-dir=<path>] [--work-tree=<path>]  
[-c name=value] [--help]  
<command> [<args>]
```

The most commonly used git commands are:

- add Add file contents to the index
- bisect Find by binary search the change that introduced a bug

...

# GIT HELP

```
$ git help config
```

GIT-CONFIG(1)

Git Manual

GIT-CONFIG(1)

*Pass in any git command*

## NAME

git-config - Get and set repository or global options

## SYNOPSIS

```
git config [<file-option>] [type] [-z|--null] name [value [value_regex]]  
git config [<file-option>] [type] --add name value
```

...

# SETTING UP GIT

```
$ git config --global user.name "Gregg Pollack"
```

*Who gets credit for changes*

```
$ git config --global user.email gregg@codeschool.com
```

*What email you use*

```
$ git config --global color.ui true
```

*Pretty command line colors*

# STARTING A REPO

```
$ mkdir store
```

```
$ cd store
```

```
$ git init
```

```
Initialized empty Git repository in /Users/gregg/store/.git/
```

*git metadata is stored here*



# GIT WORK FLOW



Jane creates README.txt file

*Starts as untracked*



Add file to staging area

*Getting ready to take a picture*



Commit changes

*A snapshot of those on the stage*

# GIT WORK FLOW



Jane creates README.txt file



Add file to staging area



Commit changes



Jane modifies README.txt file & adds LICENSE



Add both files to staging area



Commit changes



# Jane creates README.txt file

```
$ git status ← To check what's changed since last commit  
# On branch master  
#  
# Initial commit  
#  
# Untracked files:  
#   (use "git add <file>..." to include in what will be committed)  
#  
#   README.txt ← Our newly created file  
nothing added to commit but untracked files present (use "git add" to track)
```

# Add file to staging area

```
$ git add README.txt  
  
$ git status  
  
# On branch master  
#  
# Initial commit  
#  
# Changes to be committed:  
#   (use "git rm --cached <file>..." to unstage)  
#  
#       new file:   README.txt
```



*Our staged file*

# Commit changes

Commit message

what work was done?

```
$ git commit -m "Create a README."
```

```
[master abe28da] Create a README.  
1 files changed, 1 insertions(+), 0 deletions(-)  
create mode 100644 README.txt
```

```
$ git status  
# On branch master  
nothing to commit (working directory clean)
```

No new or modified files since last commit



master



# Jane modifies README.txt file & adds LICENSE

```
$ git status  
# On branch master  
# Changed but not updated:  
#  
#   modified:   README.txt  
#  
# Untracked files:  
#  
#   LICENSE  
no changes added to commit
```

master



# Add both files to staging area

```
$ git add README.txt LICENSE
```

OR

```
$ git add --all
```

← Adds all new or modified files

```
$ git status
```

```
# On branch master
# Changes to be committed:
#
#   new file:   LICENSE
#   modified:  README.txt
#
```



master

# Commit changes

master

```
$ git commit -m "Add LICENSE and finish README."
```

```
[master 1b0019c] Add LICENSE and finish README.  
2 files changed, 21 insertions(+), 0 deletions(-)  
create mode 100644 LICENSE
```



GIT  
REAL

# GIT TIMELINE HISTORY

```
$ git log
```

```
commit 1b0019c37e3f3724fb2e9035e6bab4d7d87bf455
```

```
Author: Gregg Pollack <gregg@codeschool.com>
```

```
Date: Thu Jul 5 22:31:27 2012 -0400
```

Add LICENSE and finish README.

```
commit 5acaf86b04aaaf9cbbb8ebb9042a20a46d0b9ce76
```

```
Author: Gregg Pollack <gregg@codeschool.com>
```

```
Date: Thu Jul 5 22:00:46 2012 -0400
```

Create a README.

master

# DIFFERENT WAYS TO ADD

```
$ git add <list of files>
```

Add the list of files

```
$ git add --all
```

Add all files

```
$ git add *.txt
```

Add all txt files in current directory

```
$ git add docs/*.txt
```

Add all txt files in docs directory

```
$ git add docs/
```

Add all files in docs directory

```
$ git add "* .txt"
```

Add all txt files in the whole project

# STAGING & REMOTES

LEVEL 2

# GIT DIFF



## LICENSE

Copyright (c) 2012 Envy Labs LLC

...

Permission is hereby granted,  
free of charge, to any person  
obtaining a copy

*edit*



## LICENSE

Copyright (c) 2012 Code School LLC

...

Permission is hereby granted, free  
of charge, to any person obtaining  
a copy

```
$ git diff
diff --git a/LICENSE b/LICENSE
index 7e4922d..442669e 100644
--- a/LICENSE
+++ b/LICENSE
@@ -1,4 +1,4 @@
-Copyright (c) 2012 Envy Labs LLC
+Copyright (c) 2012 Code School LLC
```

*Show unstaged differences since last commit*

*Line removed*

*Line added*

**GIT**  
REAL

# VIEWING STAGED DIFFERENCES

```
$ git add LICENSE
```

```
$ git diff
```

```
$ git diff --staged
```

```
diff --git a/LICENSE b/LICENSE
```

```
index 7e4922d..442669e 100644
```

```
--- a/LICENSE
```

```
+++ b/LICENSE
```

```
@@ -1,4 +1,4 @@
```

```
-Copyright (c) 2012 Envy Labs LLC
```

```
+Copyright (c) 2012 Code School LLC
```

No differences, since all changes are staged

View staged differences



# UNSTAGING FILES

```
$ git status  
# On branch master  
# Changes to be committed:  
#   (use "git reset HEAD <file>..." to unstage)  
#
```

```
# modified:    LICENSE
```

```
$ git reset HEAD LICENSE
```

Unstaged changes after reset:

```
M LICENSE
```

Unstage Tip

Refers to last commit

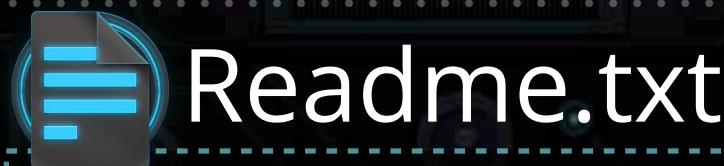


# DISCARD CHANGES

```
$ git status  
# On branch master  
# Changed but not updated:  
#   (use "git add <file>..." to update what will be committed)  
#   (use "git checkout -- <file>..." to discard changes in working directory)  
#  
# modified:    LICENSE  
#  
$ git checkout -- LICENSE  
$ git status  
nothing to commit (working directory clean)
```

*Blow away all changes  
since last commit*

# SKIP STAGING AND COMMIT



here is my readme  
and now I can sleep



here is my readme  
the cake is a lie

*edit*

*Add changes from all tracked files*

```
$ git commit -a -m "Modify readme"  
[master d00fefc] Modify readme  
1 files changed, 1 insertions(+), 1 deletions(-)
```

*Doesn't add new (untracked) files*



# UNDOING A COMMIT

Whoops, we forgot something on that commit.

```
Reset into staging
$ git reset --soft HEAD^
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified: README.txt
#
```

*Move to commit before 'HEAD'*



# ADDING TO A COMMIT

Maybe we forgot to add a file

Add to the last commit

```
$ git add todo.txt  
$ git commit --amend -m "Modify readme & add todo.txt."
```

```
[master fe98ef9] Modify readme and add todo.txt.  
2 files changed, 2 insertions(+), 1 deletions(-)  
create mode 100644 todo.txt
```

Whatever has been staged is added to last commit

# USEFUL COMMANDS

\$ git reset --soft HEAD<sup>^</sup> Undo last commit, put changes into staging

\$ git commit --amend -m "New Message"

Change the last commit

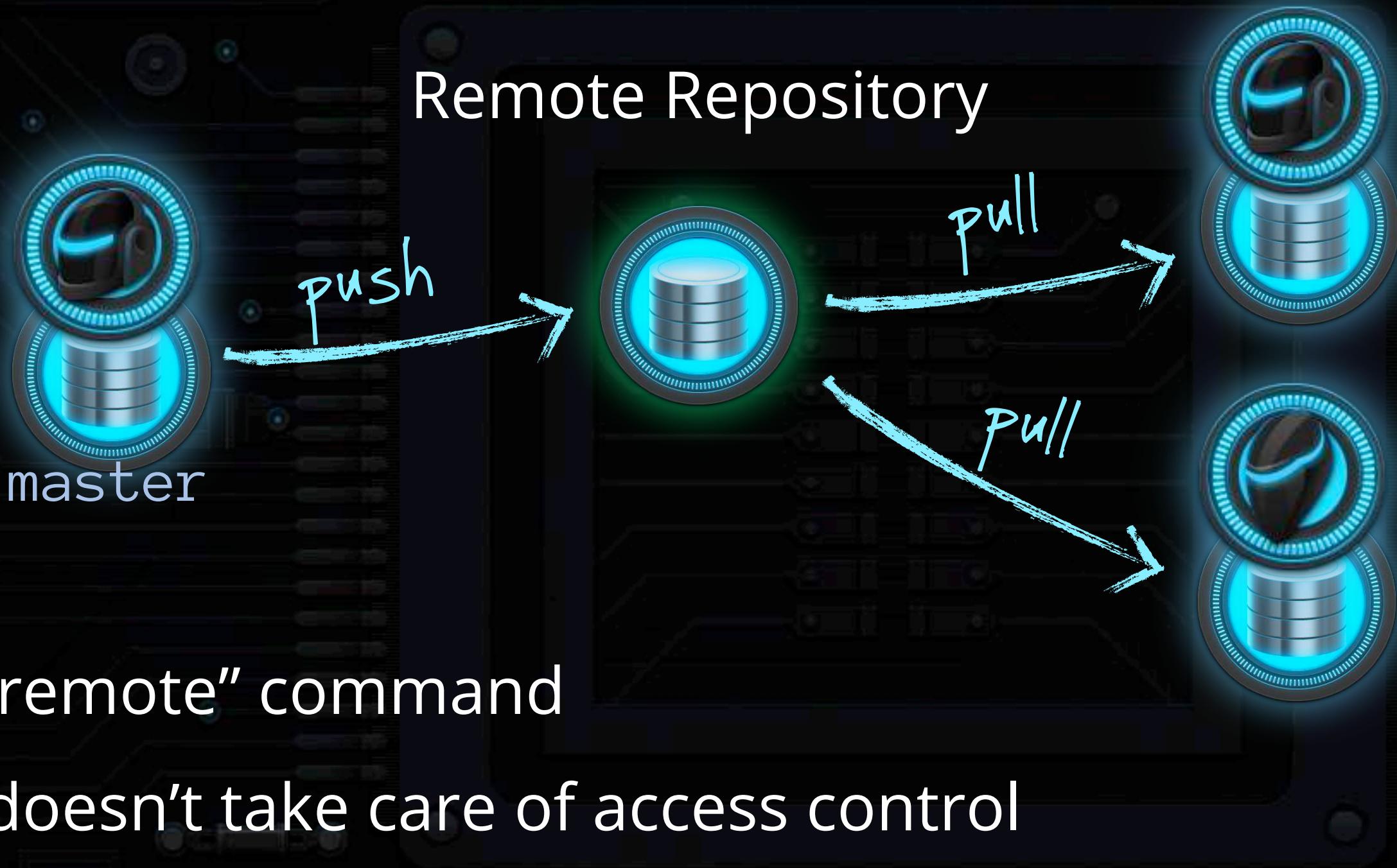
\$ git reset --hard HEAD<sup>^</sup>

Undo last commit and all changes

\$ git reset --hard HEAD<sup>^^</sup>

Undo last 2 commits and all changes

# HOW TO SHARE?



LEVEL 2 – STAGING & REMOTES

GIT  
REAL

# REMOTE REPOSITORY HOSTING

## Hosted

- GitHub
- BitBucket



master



## Self Managed

- Gitosis
- Gitorious



## Owner



Gregg

## Repository name

git-real



Great repository names are short and memorable. Need inspiration? How about [glowing-tyrion](#).

## Description (optional)

 **Public** 

Anyone can see this repository. You choose who can commit.

 **Private** 

You choose who can see and commit to this repository.

 **Initialize this repository with a README**

This will allow you to git clone the repository immediately.

Add .gitignore: **None** ▾**Create repository**

[Code](#)[Network](#)[Pull Requests 0](#)[Issues 0](#)[Wiki](#)[Graphs](#)

## Global setup:

[Set up git](#)

```
git config --global user.name "Gregg Pollack"  
git config --global user.email greggpollack@gmail.com
```

## Next steps:

```
mkdir git-real  
cd git-real  
git init  
touch README  
git add README  
git commit -m 'first commit'  
git remote add origin https://github.com/Gregg/git-real.git  
git push -u origin master
```

## Existing Git Repo?

```
cd existing_git_repo  
git remote add origin https://github.com/Gregg/git-real.git  
git push -u origin master
```

# ADDING A REMOTE

```
$ git remote add origin https://github.com/Gregg/git-real.git
```

New remote

our name for this remote

address



origin

*show remote repositories*

```
$ git remote -v
```

```
origin  https://github.com/Gregg/git-real.git (fetch)  
origin  https://github.com/Gregg/git-real.git (push)
```

# PUSHING TO REMOTE

remote repository name      local branch to push

```
$ git push -u origin master
```

Username for 'https://github.com': Gregg

Password for 'https://Gregg@github.com':

Counting objects: 11, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (6/6), done.

Writing objects: 100% (11/11), 1.50 KiB, done.

Total 11 (delta 0), reused 0 (delta 0)

To https://github.com/Gregg/git-real.git

\* [new branch]                  master -> master



push



>Password caching ➡ <https://help.github.com/articles/set-up-git>

GIT  
REAL

PUBLIC

## Gregg / git-real

Admin

Pull Request

Unwatch

1

Fork

1

Code

Network

Pull Requests 0

Issues 0

Wiki

Graphs

No description or homepage.

Clone in Mac

ZIP

HTTP

SSH

Git Read-Only

<https://github.com/Gregg/git-real.git>

Read+Write access

branch: master ▾

Files

Commits

Branches 1

Tags

Downloads

Latest commit to the master branch

Modify readme and add todo.txt.

Gregg authored 2 days ago

commit fe98ef98ed

git-real /

name	age	message	history
LICENSE	3 days ago	Add LICENSE and finish README. [Gregg]	
README.txt	2 days ago	Modify readme and add todo.txt. [Gregg]	
todo.txt	2 days ago	Modify readme and add todo.txt. [Gregg]	

## Gregg / git-real

Admin

Pull Request

Unwatch

1

Fork

1

Code

Network

Pull Requests 0

branch: master

Files

Commits

Branches 1

Tags

Downloads

## git-real / Commit History

Keyboard shortcuts available

Jul 07, 2012

**Modify readme and add todo.txt.**

Gregg authored 2 days ago

fe98ef98ed

Browse code

Jul 06, 2012

**Add LICENSE and finish README.**

Gregg authored 3 days ago

08495290b6

Browse code

Jul 05, 2012

**Create a README.**

Gregg authored 4 days ago

5acaf86b04

Browse code

Same information from "git log"

# PULLING FROM REMOTE

To pull changes down from the remote

```
$ git pull  
remote: Counting objects: 5, done.  
remote: Compressing objects: 100% (1/1), done.  
remote: Total 3 (delta 1), reused 3 (delta 1)  
Unpacking objects: 100% (3/3), done.  
From https://github.com/Gregg/git-real  
    fe98ef9..4e67ded  master      -> origin/master  
Updating fe98ef9..4e67ded  
Fast-forward  
  todo.txt | 1 +  
  1 file changed, 1 insertion(+)
```



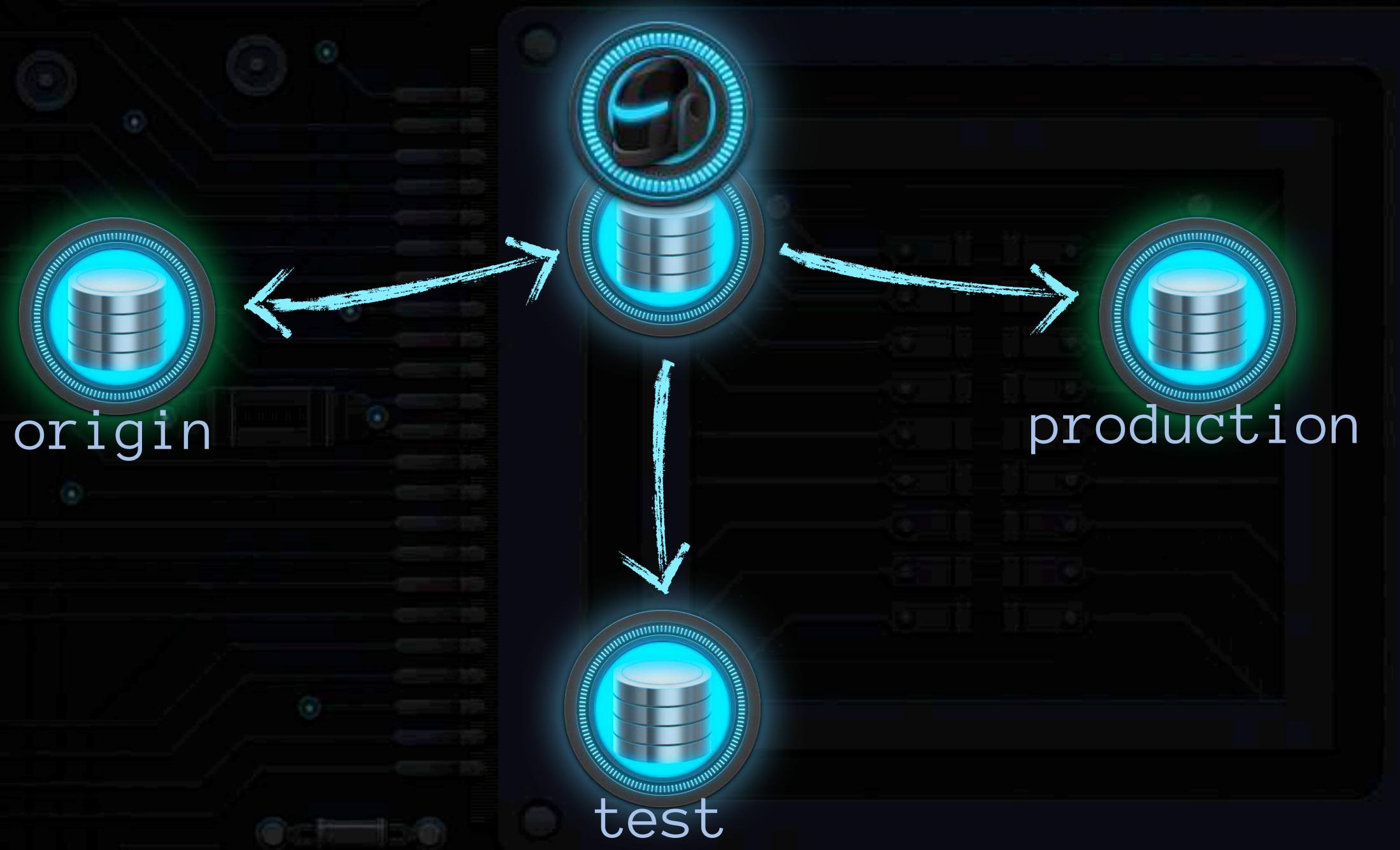
pull



origin

*It's good to do this often*

# HAVING MULTIPLE REMOTES



LEVEL 2 – STAGING & REMOTES

GIT  
REAL

# WORKING WITH REMOTES



To add new remotes

```
$ git remote add <name> <address>
```

To remove remotes

```
$ git remote rm <name>
```

To push to remotes

```
$ git push -u <name> <branch>
```



usually master

# HEROKU REMOTE

```
$ heroku create
```

Creating dev-server-1426... done, stack is cedar

http://dev-server-1426.herokuapp.com/ | git@heroku.com: dev-server-1426.git

Git remote heroku added

```
$ git remote -v
```

heroku git@heroku.com: dev-server-1426.git (fetch)

heroku git@heroku.com: dev-server-1426.git (push)

origin https://github.com/Gregg/git-real.git (fetch)

origin https://github.com/Gregg/git-real.git (push)

git repo ssh address



## To push to heroku

```
$ git push heroku master
```

→ triggers deploy

# USEFUL COMMANDS

```
$ git reset --soft HEAD^
```

```
$ git commit --amend -m "New Message"
```

```
$ git reset --hard HEAD^
```

```
$ git reset --hard HEAD^^
```

*Don't do these after you push*

# CLONING & BRANCHING

LEVEL 3



# COLLABORATING

Gregg



local repo

“Clone the repo!”

“I want a copy”

Jane

?

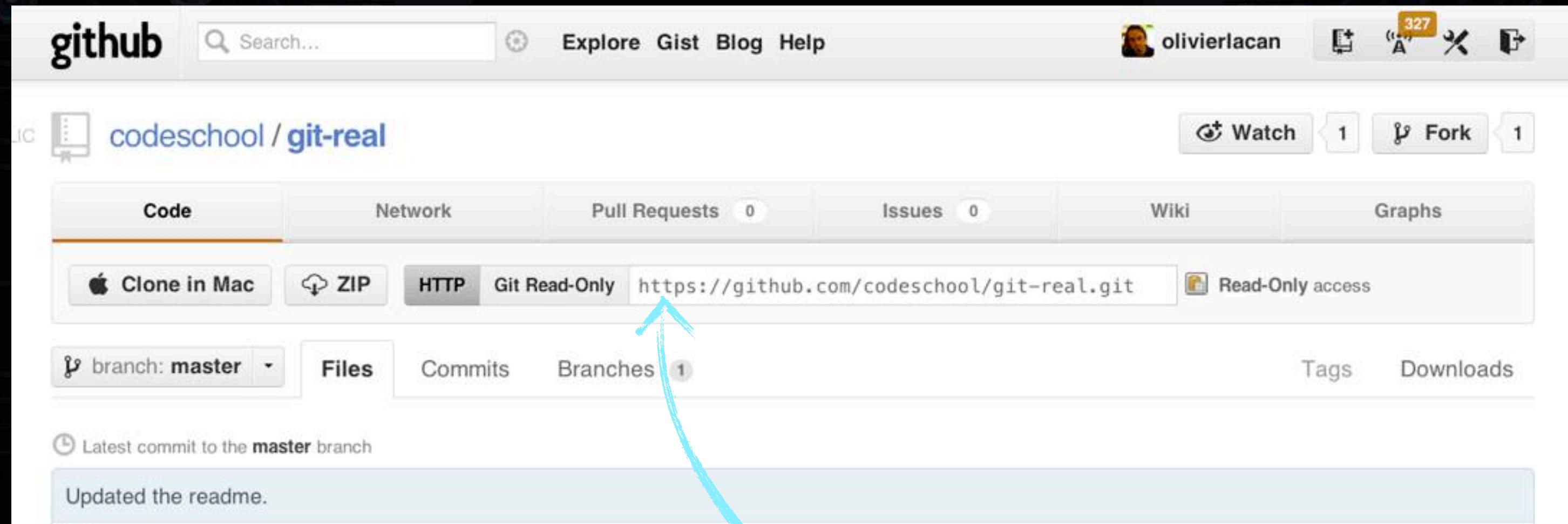
github

How do we start collaborating?

LEVEL 3 – CLONING & BRANCHING

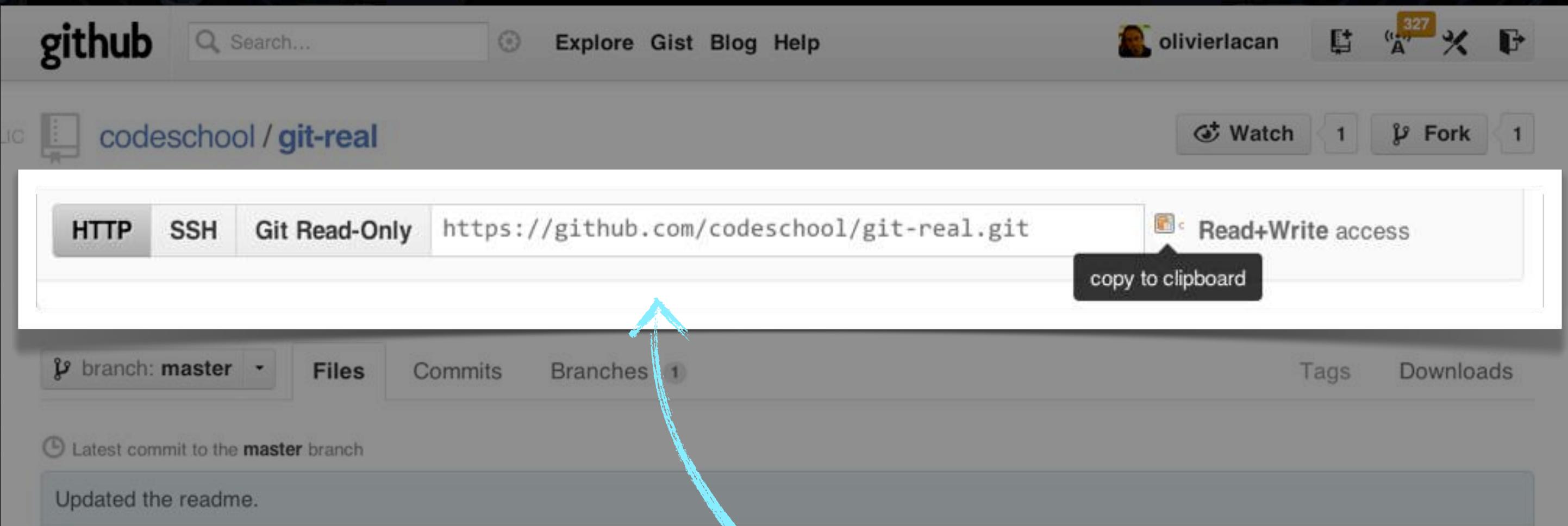
GIT  
REAL

# FINDING THE REPO URL



URL of the remote repository

# FINDING THE REPO URL



URL of the remote repository

LEVEL 3 – CLONING & BRANCHING

GIT  
REAL

# CLONING A REPOSITORY

```
$ git clone https://github.com/codeschool/git-real.git  
Cloning into 'git-real'...
```

URL of the remote repository

```
$ git clone https://github.com/codeschool/git-real.git git-demo  
Cloning into 'git-demo'...
```

local folder name

# GIT CLONE

1 - Downloads the entire repository into a new git-real directory.

2 - Adds the 'origin' remote, pointing it to the clone URL.

```
$ git remote -v
```

```
origin  https://github.com/codeschool/git-real.git (fetch)
origin  https://github.com/codeschool/git-real.git (push)
```

3 - Checks out initial branch (likely master).

sets the head

# BRANCHING OUT

Need to work on a feature that will take some time?  
Time to branch out.

```
$ git branch cat
```

```
$ git branch  
cat  
* master
```

branch created from master

HEAD still on master

Jane

cat

HEAD

master

# SWITCHING TO A BRANCH

Time to jump on that new 'cat' branch.

```
$ git checkout cat
```

```
Switched to branch 'cat'
```

HEAD is now on 'cat'

Jane

cat

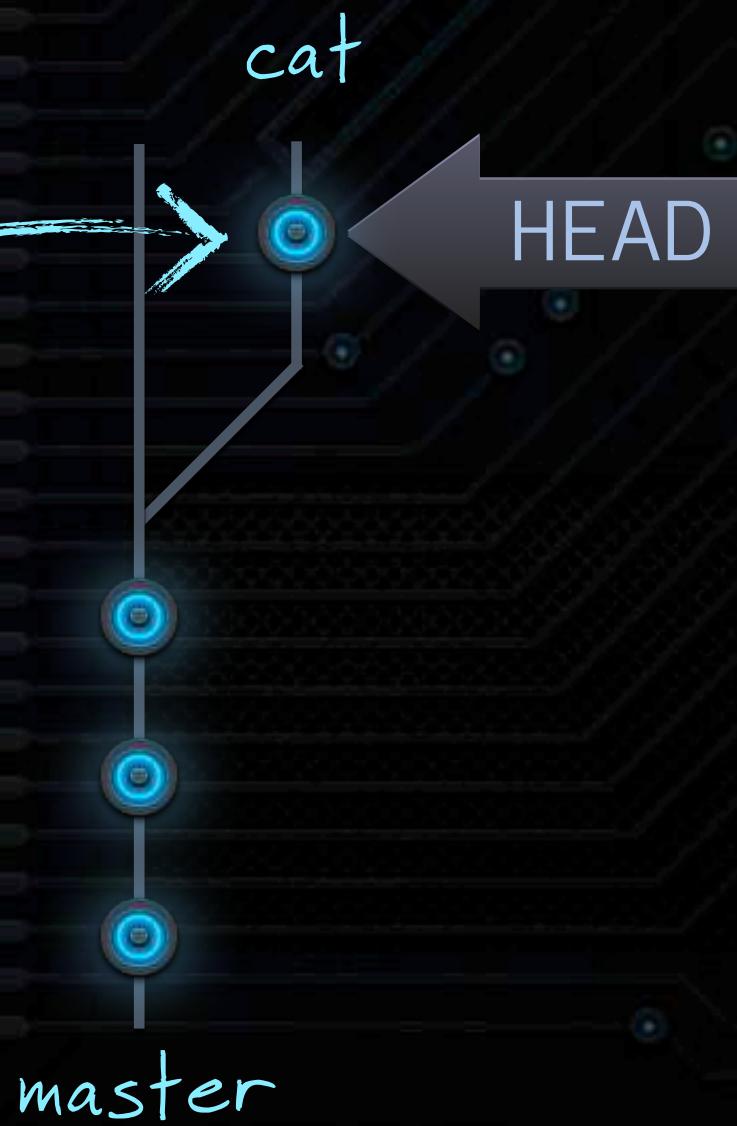
HEAD

master

# WORKING ON A BRANCH

```
$ echo "Schrödinger" > cat.txt  
$ git add cat.txt  
$ git commit -m "Create quantum cat."  
[cat ab48a3f] Create quantum cat.  
1 file changed, 1 insertion(+)  
create mode 100644 cat.txt
```

committed to the  
"cat" branch

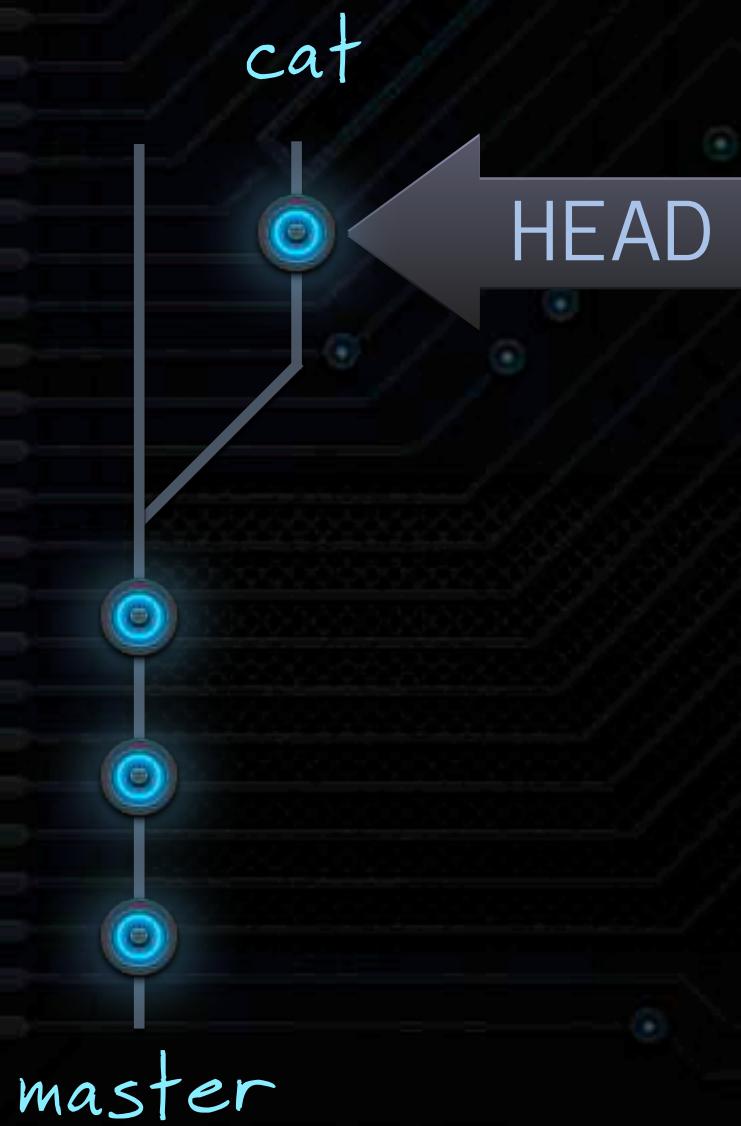


# WORKING ON A BRANCH

```
$ ls  
README.txt cat.txt
```



see the cat?



# BACK TO MASTER

```
$ git checkout master
```

```
Switched to branch 'master'
```

```
$ ls
```

```
README.txt
```

```
$ git log
```

```
commit 1191ceb7252c9d4b1e05c9969a55766a8adfce3b
```

```
Author: Gregg <gregg@codeschool.com>
```

```
Date: Wed Jun 27 23:11:20 2012 -0700
```

Add README.

*nothing in the log either*

LEVEL 3 – CLONING & BRANCHING

*cat.txt is gone!*

*and we're back on master*

HEAD



GIT  
REAL

# BACK TO CAT

```
$ git checkout cat  
Switched to branch 'cat'  
$ ls  
README.txt cat.txt
```

phew, still here



# TIME TO MERGE

Done with that feature branch? Time to merge it into 'master'.

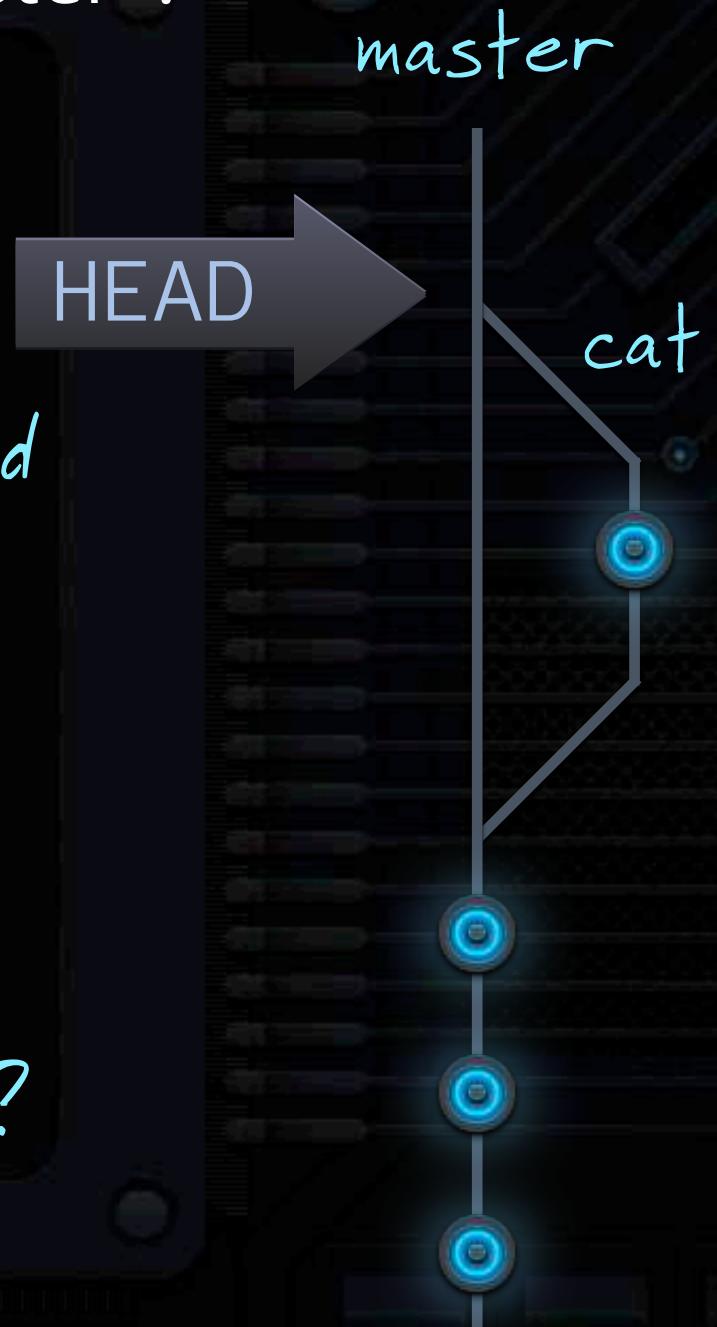
```
$ git checkout master  
Switched to branch 'master'  
$ ls  
README.txt
```

no cat, as expected

```
$ git merge cat  
Updating 1191ceb..ab48a3f  
Fast-forward  
 cat.txt | 1 +  
 1 file changed, 1 insertion(+)  
 create mode 100644 cat.txt
```

what's that?

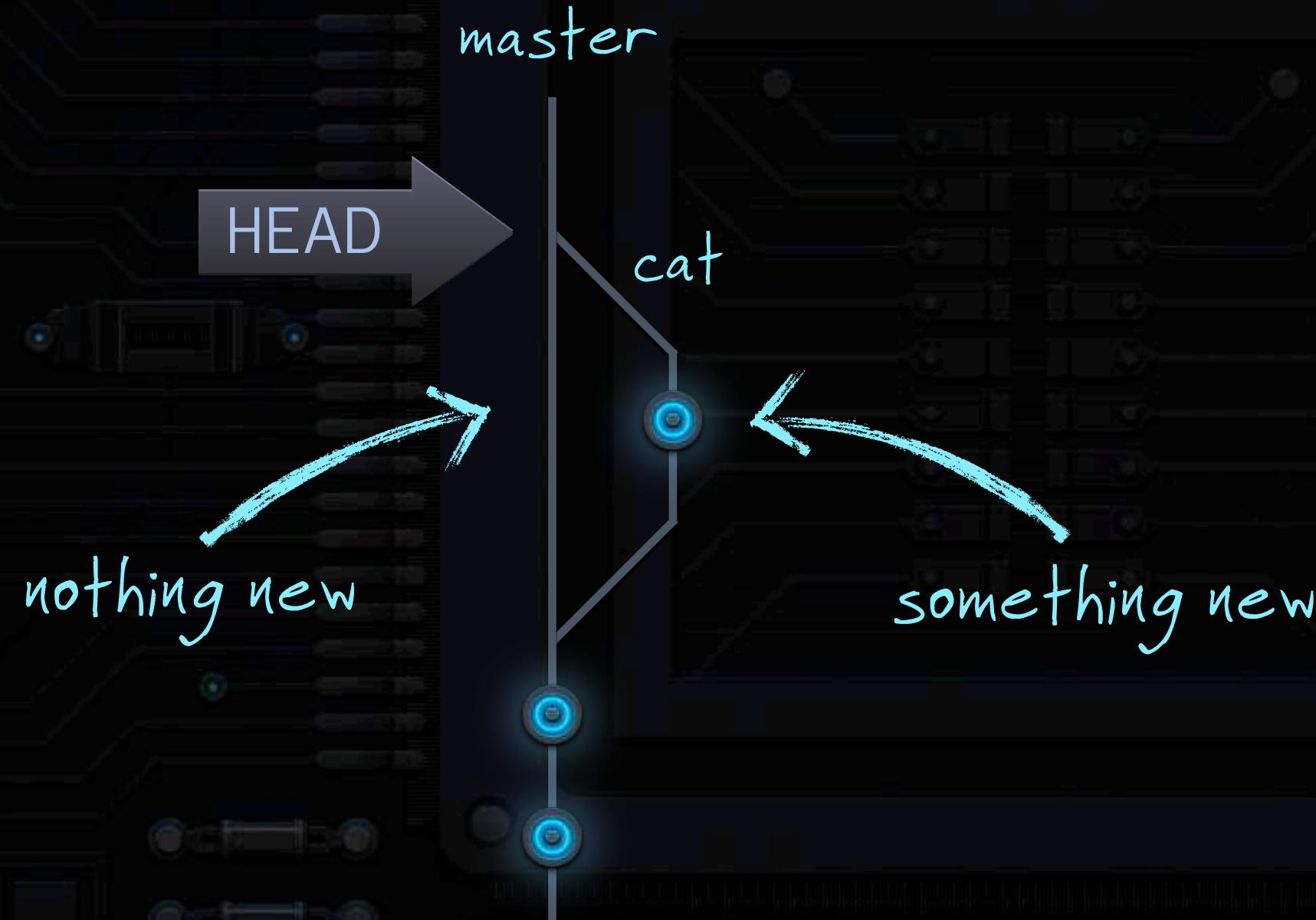
merge brings one branch's changes into another



GIT  
REAL

# FAST-FORWARD TO THE FUTURE

Conditions for a fast-forward merge



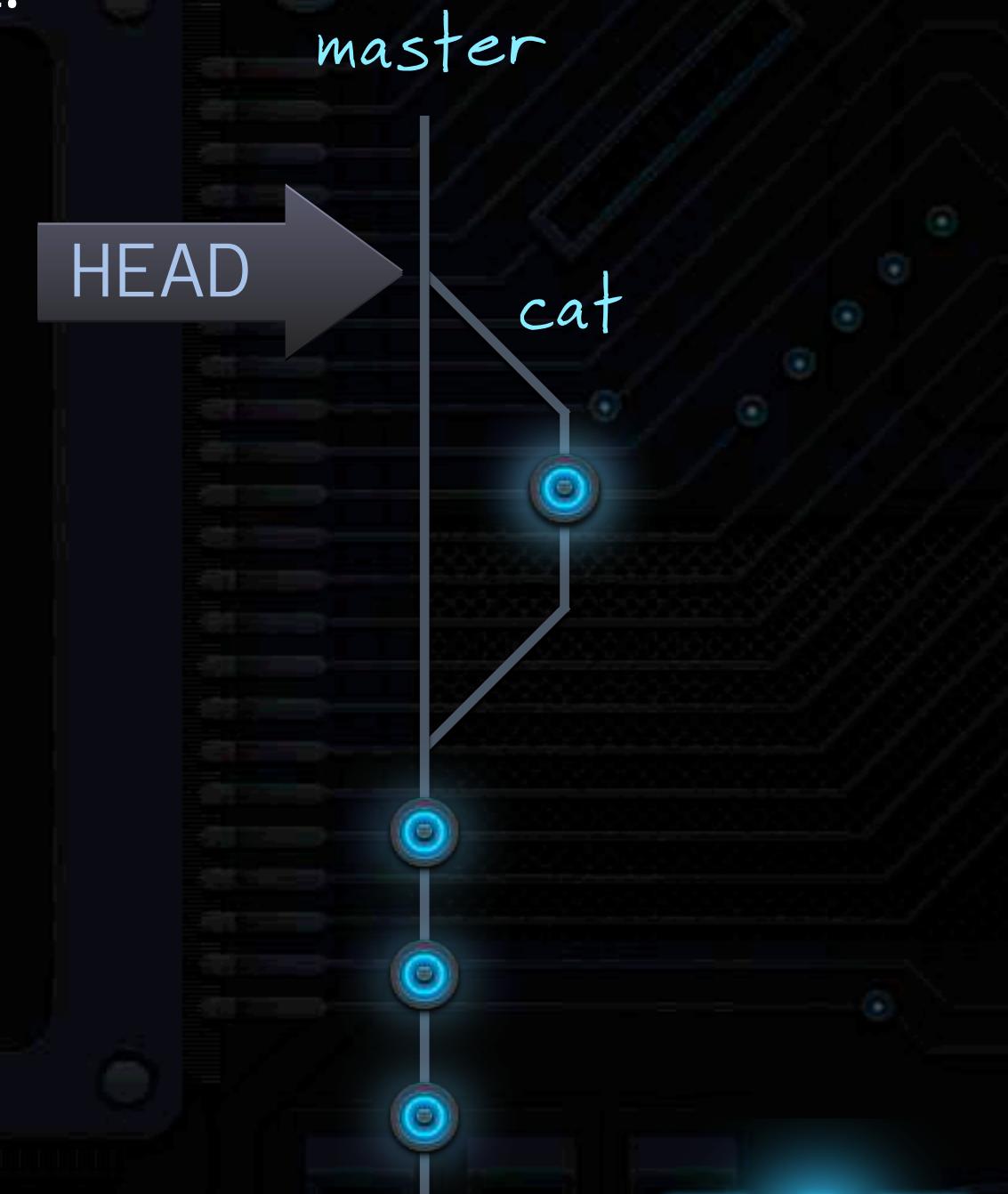
LEVEL 3 – CLONING & BRANCHING

GIT  
REAL

# BRANCH CLEAN UP

When you're done with a branch, you can safely remove it.

```
$ git branch -d cat  
Deleted branch cat (was 957dbff).
```



# NON-FAST-FORWARD

Let's work on a new admin feature.

```
$ git checkout -b admin  
Switched to a new branch 'admin'
```

```
...  
$ git add admin/dashboard.html  
$ git commit -m 'Add dashboard'
```

```
...  
$ git add admin/users.html  
$ git commit -m 'Add user admin'
```

*creates and checks out branch*

master

admin

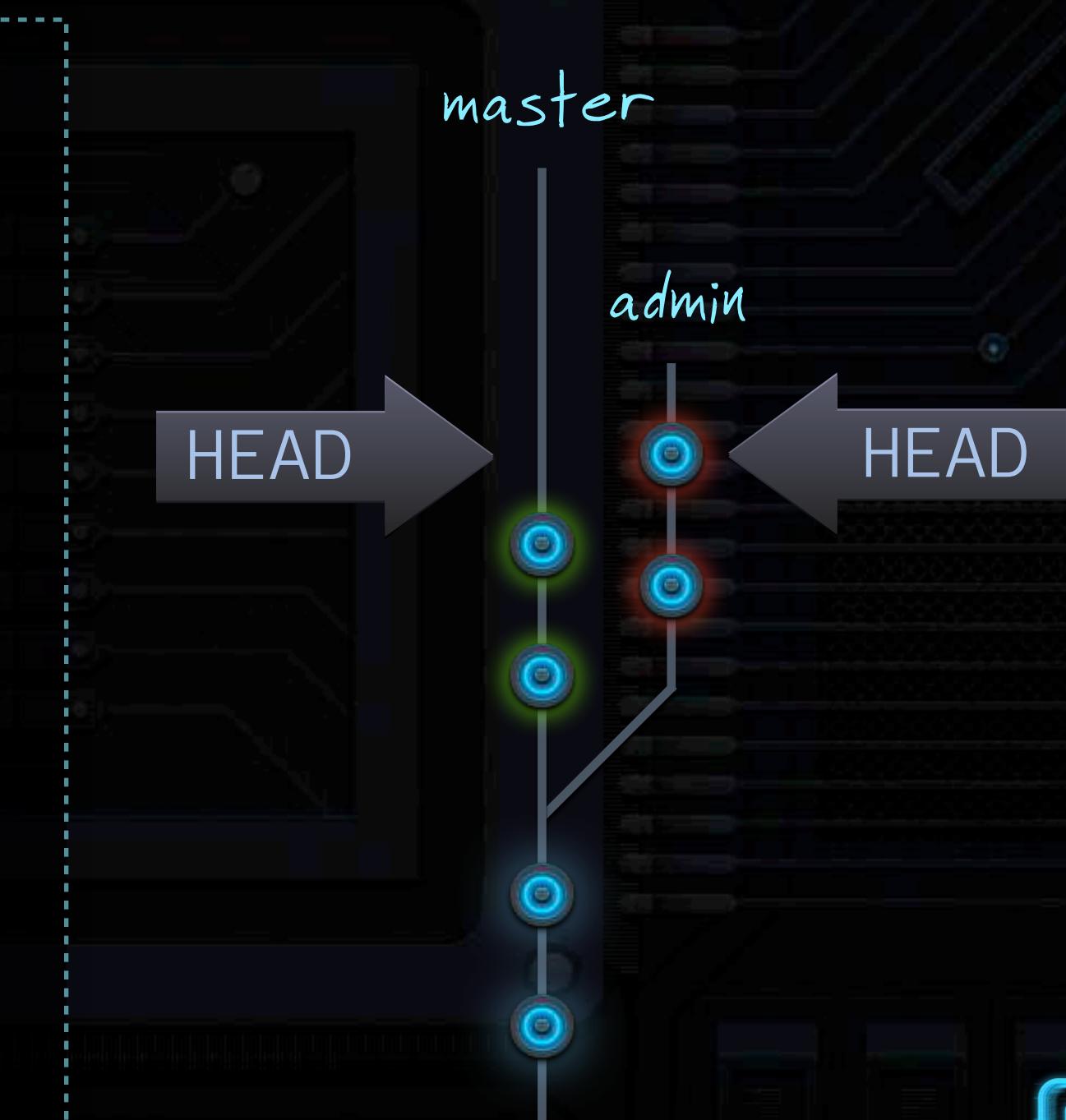
HEAD

“Please fix the bugs on master.”

# BUG FIXING ON MASTER

Time to put out the fire. We'll get back to that admin branch later.

```
$ git checkout master  
Switched to branch 'master'  
$ git branch  
admin  
* master  
$ git pull  
  
$ git add store.rb  
$ git commit -m 'Fix store bug'  
  
$ git add product.rb  
$ git commit -m 'Fix product'  
  
$ git push
```

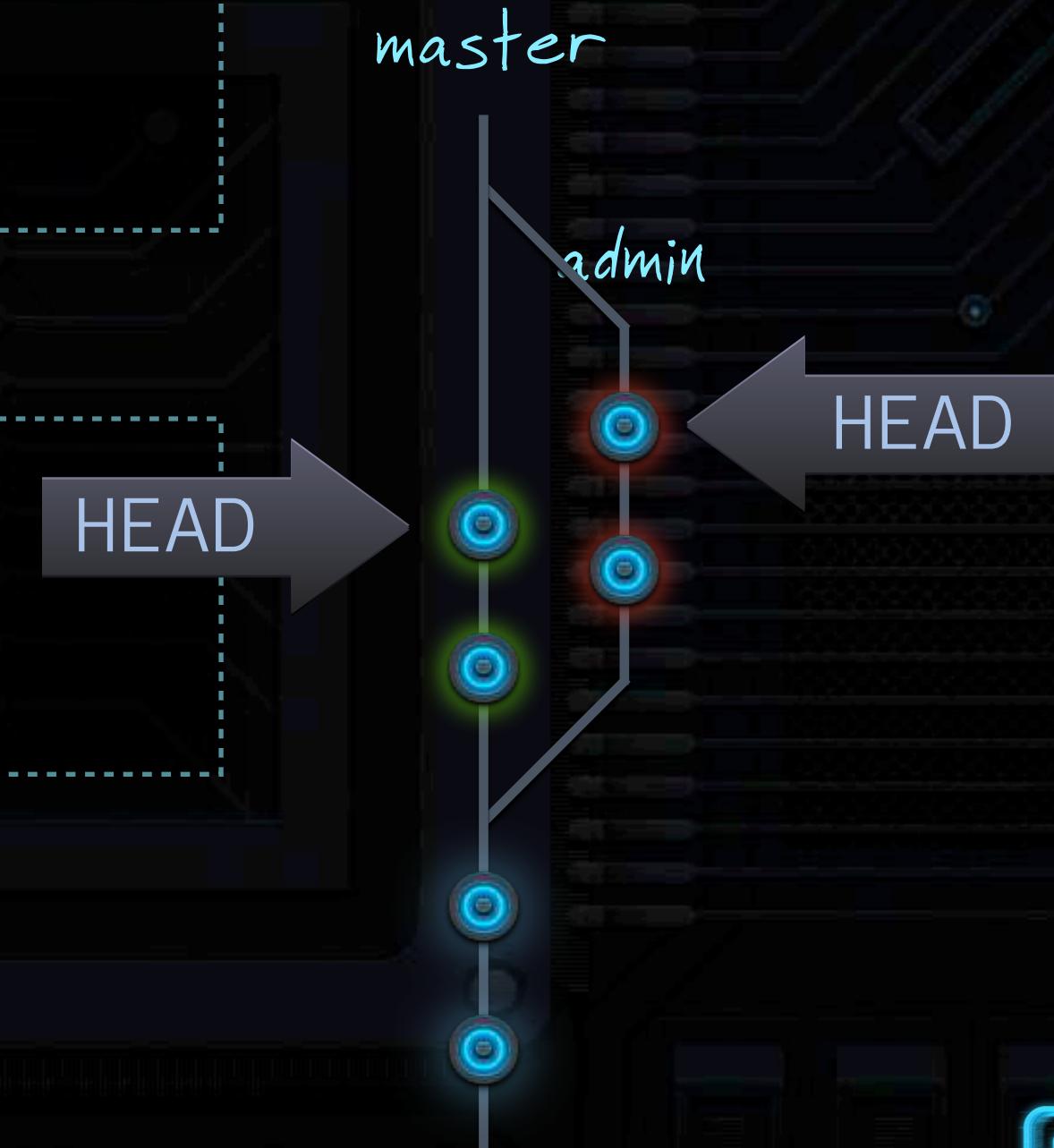


# BACK TO OUR BRANCH

```
$ git checkout admin  
Switched to branch 'admin'
```

When ready to bring in changes

```
$ git checkout master  
Switched to branch 'admin'  
$ git merge admin
```



# AND SUDDENLY...

Git uses Vi if no default editor is set to edit commit messages.

```
1 Merge branch 'admin'  
2  
3 # Please enter a commit message to explain why this merge is necessary,  
4 # especially if it merges an updated upstream into a topic branch.  
5 #  
6 # Lines starting with '#' will be ignored, and an empty message aborts  
7 # the commit.
```

:wq + hit Enter to write (save) & quit

## Vi commands

j down    k up    ESC leave mode    :wq save & quit

. h left    l right    i insert mode    :q! cancel & quit

**DON'T PANIC**

**GIT**  
REAL

# RECURSIVE MERGING

Git can't fast-forward since changes were made in both branches.

Merge made by the 'recursive' strategy.

0 files changed

create mode 100644 admin/dashboard.html

create mode 100644 admin/users.html

A commit was created to merge the two branches.

```
$ git log
```

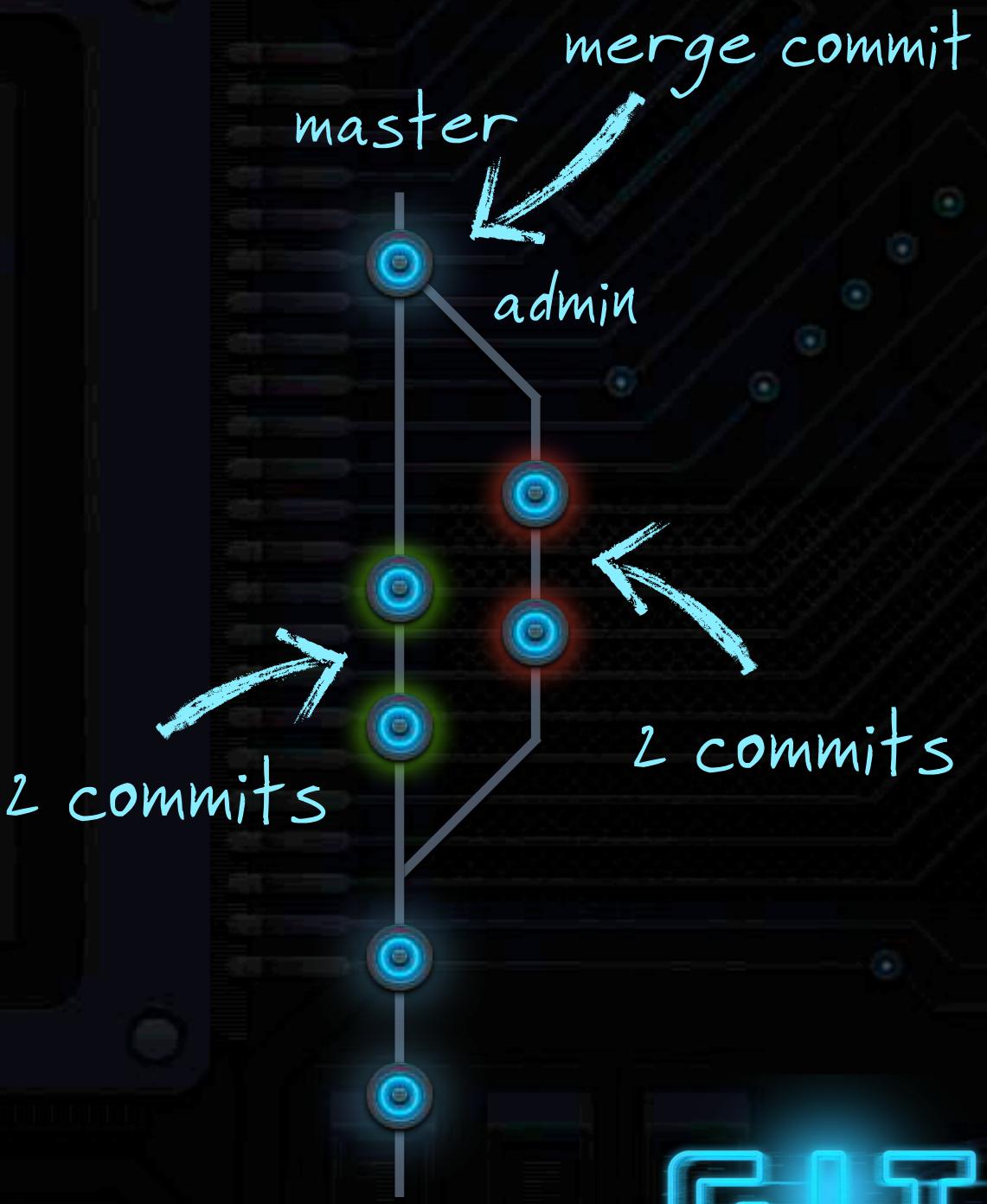
```
commit 19f735c3556129279bb10a0d1447dc5aba1e1fa9
```

```
Merge: 5c9ed90 7980856
```

```
Author: Jane <Jane@CodeSchool.com>
```

```
Date: Thu Jul 12 17:51:53 2012 -0400
```

Merge branch 'admin'



# COLLABORATION BASICS

LEVEL 4

Gregg

push

github

Clone the repo

```
$ git clone https://github.com/codeschool/git-real.git
```

Start making changes

Jane adds two files



?



Jane

I want a copy

# TWO NEW FILES

```
jane $ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       product.rb      store.rb
nothing added to commit but untracked files present
jane $ git add --all
jane $ git commit -m "Add store and product models."
[master 30ce481] Add product and store models.
2 files changed, 2 insertions(+)
create mode 100644 product.rb
create mode 100644 store.rb
```



# COMMIT FLOW

jane \$ git push

Sends her new commit

```
Counting objects: 5, done.  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (4/4), 441 bytes, done.  
Total 4 (delta 0), reused 0 (delta 0)  
To https://github.com/codeschool/git-real.git  
 4e67ded..30ce481 master -> master
```



master



push new commits

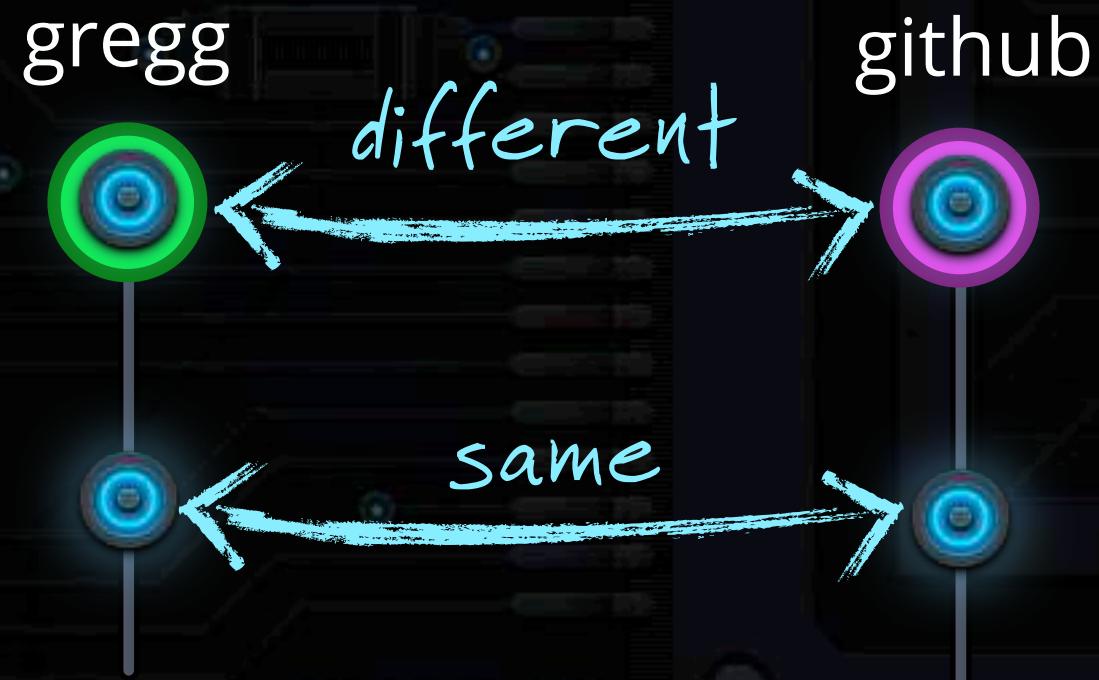


github

# DIFFERENT GIT COMMITS

```
gregg $ git commit -am "Update the readme."  
[master c715339] Update the readme.  
 1 file changed, 1 insertion(+), 1 deletion(-)
```

What happens now?



# GIT PUSH REJECTED

```
gregg $ git push
```

```
To https://github.com/codeschool/git-real.git
```

```
! [rejected]          master -> master (non-fast-forward)
```

```
error: failed to push some refs to 'https://github.com/codesch...-real.git'
```

```
hint: Updates were rejected because the tip of your current branch is behind
```

```
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
```

```
hint: before pushing again.
```

```
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
$ git pull
```

```
...  
$ git push Success!
```

LEVEL 4 – COLLABORATION BASICS

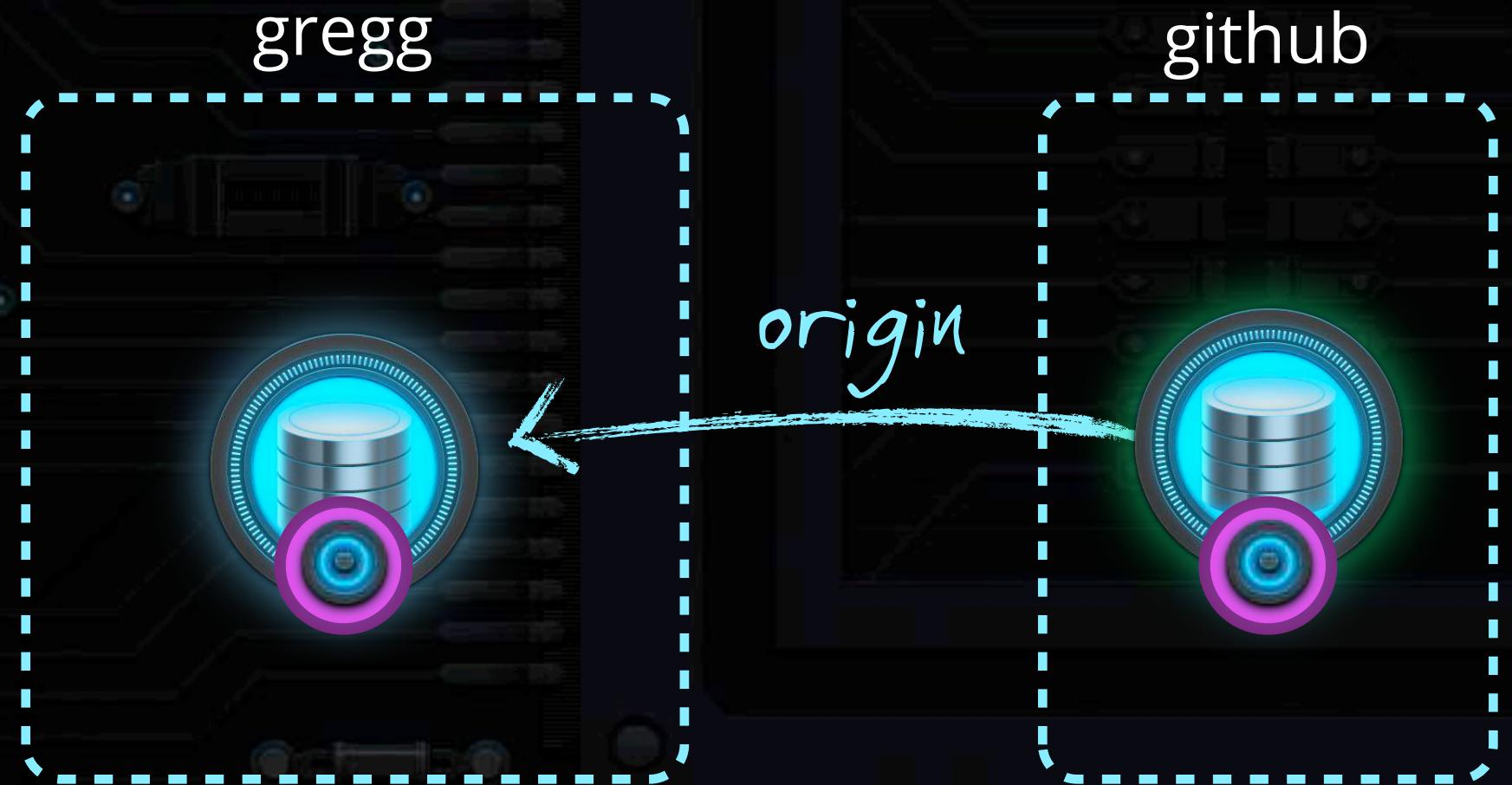
CODESCHOOL

GIT  
REAL

# UNDERSTANDING PULL

```
$ git pull
```

1. Fetch (or Sync) our local repository with the remote one



```
$ git fetch
```

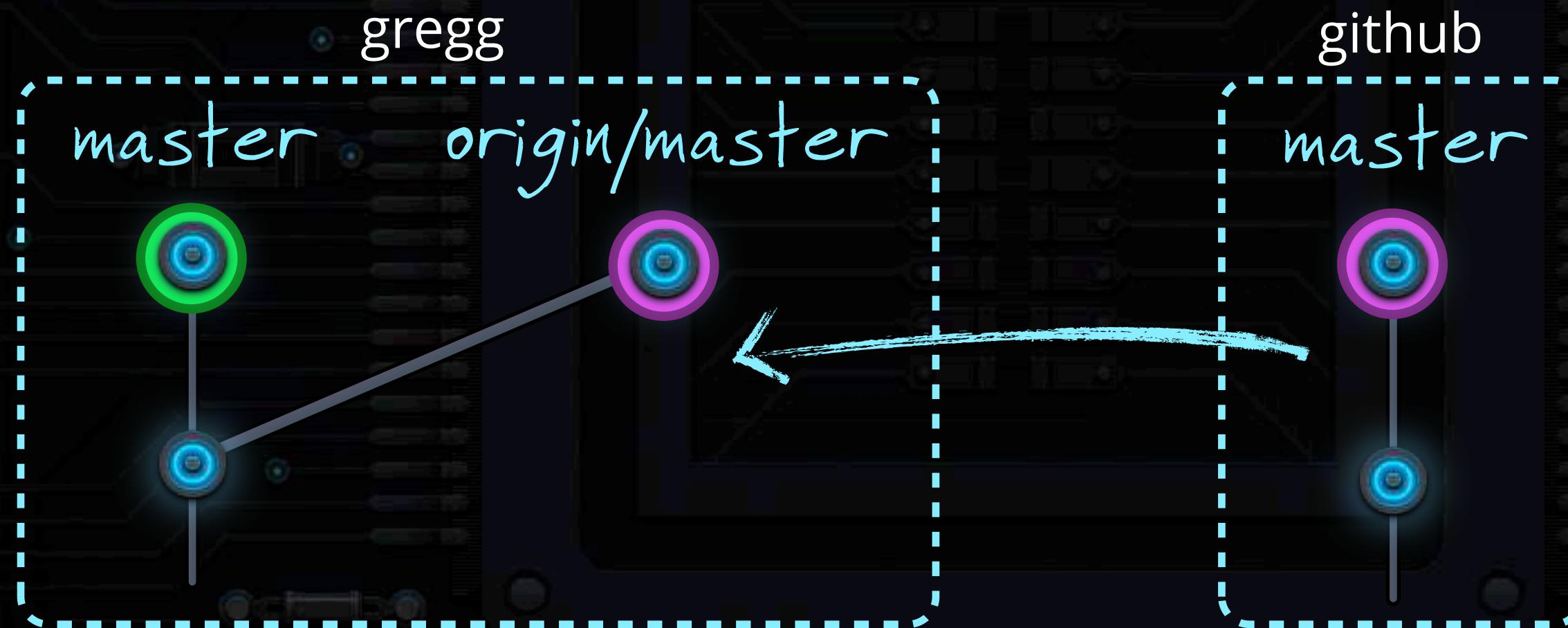
Fetch doesn't actually update any of our local code

GIT  
REAL

# UNDERSTANDING PULL

```
$ git pull
```

1. Fetch (or Sync) our local repository with the remote one



```
$ git fetch
```

2. Merges the origin/master with master

```
$ git merge origin/master
```

# MERGE COMMIT

```
$ git pull
```

1. Fetch (or Sync) our local repository with the remote one

2. Merges the origin/master with master

```
$ git merge origin/master
```

*Create a new commit for this merge*

*my editor*

```
1 Merge branch 'master' of https://github.com/codeschool/git-real
2
3 # Please enter a commit message to explain why this merge is necessary,
4 # especially if it merges an updated upstream into a topic branch.
5 #
6 # Lines starting with '#' will be ignored, and an empty message aborts
7 # the commit.
```

# GIT PUSH REJECTED

```
$ git pull
```

```
remote: Counting objects: 5, done.  
remote: Compressing objects: 100% (2/2), done.  
remote: Total 4 (delta 0), reused 4 (delta 0)  
Unpacking objects: 100% (4/4), done.  
From https://github.com/codeschool/git-real  
  4e67ded..30ce481  master      -> origin/master  
Merge made by the 'recursive' strategy.  
 product.rb | 1 +  
 store.rb   | 1 +  
 2 files changed, 2 insertions(+)  
 create mode 100644 product.rb  
 create mode 100644 store.rb
```



merge commit

GIT  
REAL

# MERGE COMMIT

```
$ git pull
```

1. Fetch (or Sync) our local repository with the remote one
2. Merges the origin/master with master

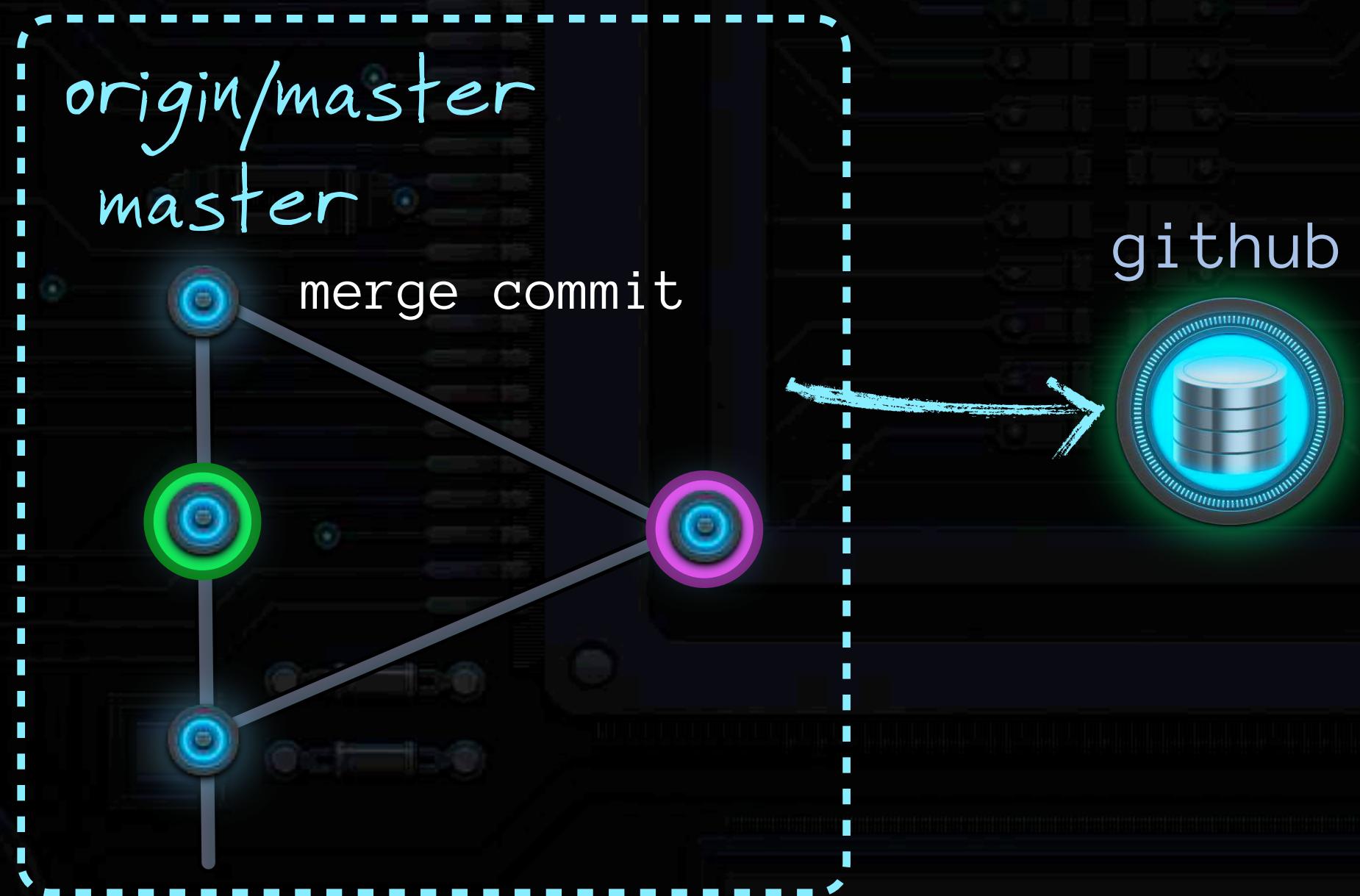


```
$ git fetch  
$ git merge origin/master
```

# PUSHING COMMITS

```
$ git push
```

Update origin/master be at the same state as our local repo



# OUR LOG

```
gregg $ git log
```

```
commit ee47baaedcd54e1957f86bda1aaa1b8a136185da  
Merge: 87c5243 57501d5  
Author: Gregg Pollack <Gregg@CodeSchool.com>
```

```
Merge branch 'master' of https://github.com/Gregg/git-real
```

```
commit 87c5243d2266f05cd9fda8b1c9137f11b3fe6f31  
Author: Gregg Pollack <Gregg@CodeSchool.com>
```

```
Update the readme.
```

```
commit 57501d595b16e2d1198a9c04c547a5b1380a6618  
Author: Gregg Pollack <Gregg@CodeSchool.com>
```

```
Add store and product models.
```

The problem with pull



# MERGE CONFLICTS



README

here is my readme

the cake is a lie



Gregg



README

here is my readme



Jane

*committed*

*committed & pushed*

gregg



*different*

github



*same*

GIT  
REAL

# MERGE CONFLICT

```
gregg $ git pull  
remote: Counting objects: 5, done.  
remote: Compressing objects: 100% (1/1), done.  
remote: Total 3 (delta 1), reused 3 (delta 1)  
Unpacking objects: 100% (3/3), done.  
From https://github.com/Gregg/git-real  
  ee47baa..4e76d35  master      -> origin/master  
Auto-merging README.txt  
CONFLICT (content): Merge conflict in README.txt  
Automatic merge failed; fix conflicts and then commit the result.
```



Git modified this file with the diff



# MERGE CONFLICT

```
gregg $ git status  
# On branch master  
# Your branch and 'origin/master' have diverged,  
# and have 1 and 1 different commit each, respectively.  
#  
# Unmerged paths:  
#   (use "git add/rm <file>..." as appropriate to mark resolution)  
#  
# both modified: README.txt ← Need to edit these files  
#  
no changes added to commit (use "git add" and/or "git commit -a")
```

# MERGE CONFLICT



## README

here is my readme

<<<<< HEAD  
the cake is a lie.

-----

the cake is telling the truth!

>>>>>

4e76d3542a7eee02ec516a47600002a90a4e4b48

Edit file and correct

Our local version

Jane's version

gregg \$ git commit -a

Merge commit

# COMMIT EDITOR

```
gregg $ git commit -a
```



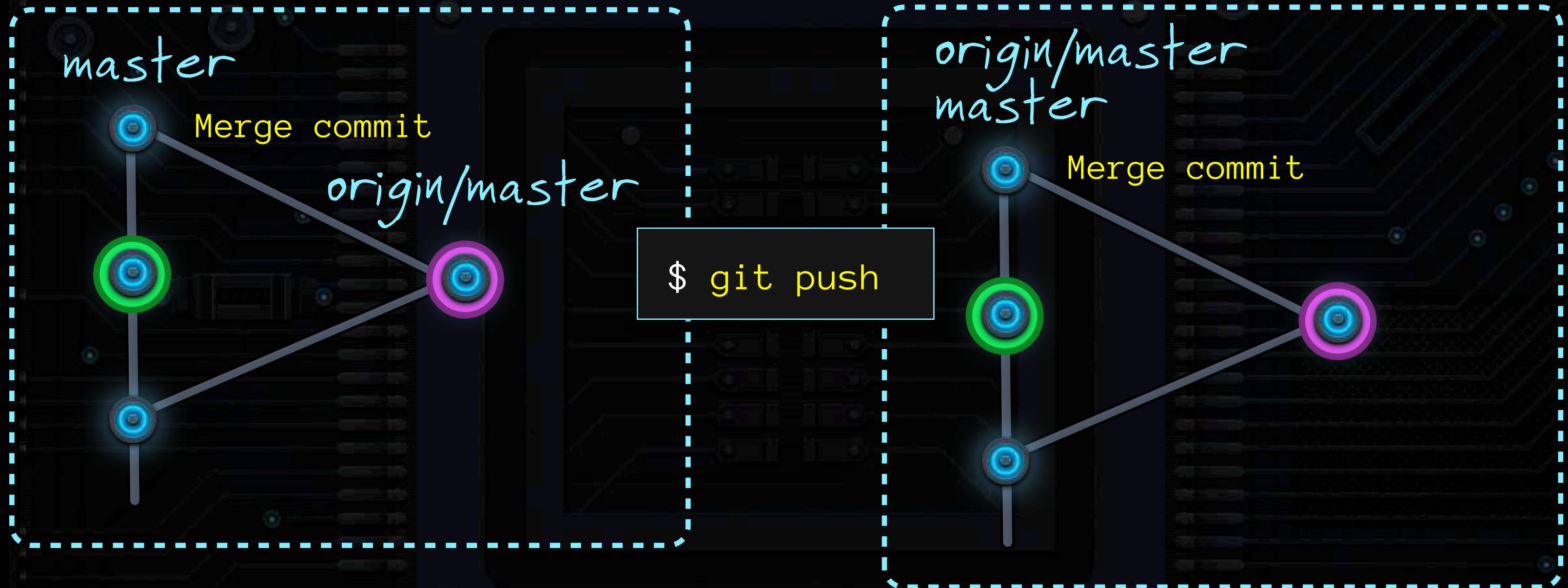
Merge commit

Editor

```
1 Merge branch 'master' of https://github.com/Gregg/git-real
2
3 Conflicts:
4 ▶ README.txt
5 #
6 # It looks like you may be committing a merge.
7 # If this is not correct, please remove the file
8 #▶.git/MERGE_HEAD
9 # and try again.
10
11
12 # Please enter the commit message for your changes. Lines starting
13 # with '#' will be ignored, and an empty message aborts the commit.
14 # On branch master
15 # Your branch and 'origin/master' have diverged,
16 # and have 1 and 1 different commit each, respectively.
```

GIT  
REAL

# MERGE COMMIT



GIT  
REAL

# REMOTE BRANCHES & TAGS

LEVEL 5

# WHY CREATE A REMOTE BRANCH?

- When you need other people to work on your branch.
- Any branch that will last more than a day.



*Gregg*

# CREATING A REMOTE BRANCH



```
$ git checkout -b shopping_cart  
Switched to a new branch 'shopping_cart'  
$ git push origin shopping_cart  
Counting objects: 10, done.  
Delta compression using up to 4 threads.  
Compressing objects: 100% (6/6), done.  
Writing objects: 100% (6/6), 619 bytes, done.  
Total 6 (delta 2), reused 0 (delta 0)  
To https://github.com/codeschool/git-real.git  
 * [new branch] shopping_cart -> shopping_cart
```

Links local branch to  
the remote branch  
(tracking)

# PUSHING TO THE BRANCH



```
$ git add cart.rb  
$ git commit -a -m "Add basic cart ability."  
[shopping_cart 2a0dbf9] Add basic cart ability  
 1 file changed, 1 insertion(+)  
 create mode 100644 cart.rb  
  
$ git push  
Counting objects: 4, done.  
Delta compression using up to 4 threads.  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (3/3), 302 bytes, done.  
Total 3 (delta 1), reused 0 (delta 0)  
To https://github.com/codeschool/git-real.git  
 786d7a1..2a0dbf9  shopping_cart -> shopping_cart
```

*Pushed changes from branch*



branch: **master**

Files

Commits

Branches 2

Tags

## Switch branches/tags



Filter branches/tags

**Branches**

**Tags**

✓ master

shopping\_cart

LICENSE

3 days ago

Add LICENSE and finish README. [Gregg]

README.txt

an hour ago

Fix README and Cake [Gregg]

product.rb

2 hours ago

Add store and product models. [Gregg]

store.rb

2 hours ago

Add store and product models. [Gregg]

todo.txt

a day ago

Add new todo item. [Gregg]

⌚ Latest commit to the **shopping\_cart** branch

Add basic cart ability



Gregg authored 23 minutes ago



commit 2

## git-real /

name	age	message
LICENSE	5 days ago	Add LICENSE and finish README. [Gregg]
README.txt	an hour ago	Fix README and Cake [Gregg]
cart.rb	23 minutes ago	Add basic cart ability [Gregg]
product.rb	2 hours ago	Add store and product models. [Gregg]
store.rb	2 hours ago	Add store and product models. [Gregg]
todo.txt	a day ago	Add new todo item. [Gregg]

# CREATING A BRANCH

Hey Jane, I started a branch

Sweet, I'll check it out

# PULLING NEW BRANCHES

```
$ git pull
remote: Counting objects: 13, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 3), reused 8 (delta 2)
Unpacking objects: 100% (9/9), done.
From https://github.com/Gregg/git-real
  4e76d35..786d7a1  master      -> origin/master
* [new branch]      shopping_cart -> origin/shopping_cart
Updating 4e76d35..786d7a1
Fast-forward
 README.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

# PULLING NEW BRANCHES

```
$ git branch
```

```
* master
```

```
$ git branch -r list all remote branches
```

```
origin/master
```

```
origin/shopping_cart
```

```
$ git checkout shopping_cart
```

Branch shopping\_cart set up to track remote branch shopping\_cart from origin.  
Switched to a new branch 'shopping\_cart'

```
$ git branch
```

```
master
```

```
* shopping_cart
```

Now we can contribute and push!

# REMOTE SHOW

```
$ git remote show origin

* remote origin
  Fetch URL: https://github.com/Gregg/git-real.git
  Push  URL: https://github.com/Gregg/git-real.git
HEAD branch: master
Remote branches:
  master            tracked
  shopping_cart    tracked
Local branches configured for 'git pull':
  master           merges with remote master
  shopping_cart   merges with remote shopping_cart
Local refs configured for 'git push':
  master           pushes to master          (up to date)
  shopping_cart   pushes to shopping_cart (local out of date)
```



# MOVING A BRANCH

```
$ git push origin :shopping_cart  Deletes remote branch  
To https://github.com/codeschool/git-real.git  
- [deleted]           shopping_cart
```

```
$ git branch -d shopping_cart  Must delete local branch manually  
error: The branch 'shopping_cart' is not fully merged.  
If you are sure you want to delete it, run 'git branch -D shopping_cart'.  
$ git branch -D shopping_cart  
Deleted branch shopping_cart (was ea0a1b9).
```

# WHAT ABOUT GREGG?



Jane



Gregg

LEVEL 5 – REMOTE BRANCHES AND TAGS

GIT  
REAL

# ON DELETED REMOTE BRANCH

```
$ git commit -m -a "Add ability to pay."
```

[shopping\_cart 9851887] Add ability to pay.

1 file changed, 1 insertion(+), 1 deletion(-)

```
$ git push
```

Everything up-to-date

No remote to push to (it's just a local branch now)

```
$ git remote show origin
```

Remote branches:

master

tracked

refs/remotes/origin/shopping\_cart stale (use 'git remote prune' to remove)

```
$ git remote prune origin
```

Pruning origin

URL: <https://github.com/codeschool/git-real.git>

\* [pruned] origin/shopping\_cart

Gregg



# REMOTE BRANCH NAMES

```
$ git branch  
* staging  
  master
```

```
$ git push heroku-staging staging
```

Would not work, would push to staging

```
$ git push heroku-staging staging:master
```

Will push and deploy staging on heroku

Heroku deploys only master branch



heroku-staging

*local:remote*

# TAGGING

A tag is a reference to a commit (used mostly for release versioning)

```
$ git tag      list all tags
```

v0.0.1

v0.0.2

```
$ git checkout v0.0.1    check out code at commit
```

To add a new tag

```
$ git tag -a v0.0.3 -m "version 0.0.3"
```

To push new tags

```
$ git push --tags
```

branch: **master**

Files

Commits

Branches 2

Tags

## Switch branches/tags

Filter branches/tags

**Branches**

**Tags**

master

shopping\_cart

LICENSE

README.txt

product.rb

store.rb

todo.txt

3 days ago

an hour ago

2 hours ago

2 hours ago

a day ago

## Switch branches/tags

Filter branches/tags

**Branches**

**Tags**

v0.0.3

v0.0.2

v0.0.1

Add store and product models. [Gregg]

Add new todo item. [Gregg]

tag: v0.0.1 ▾

Files

Commits

Branches 2

🕒 Latest commit to the v0.0.1 tag



Gregg authored 3 hours ago

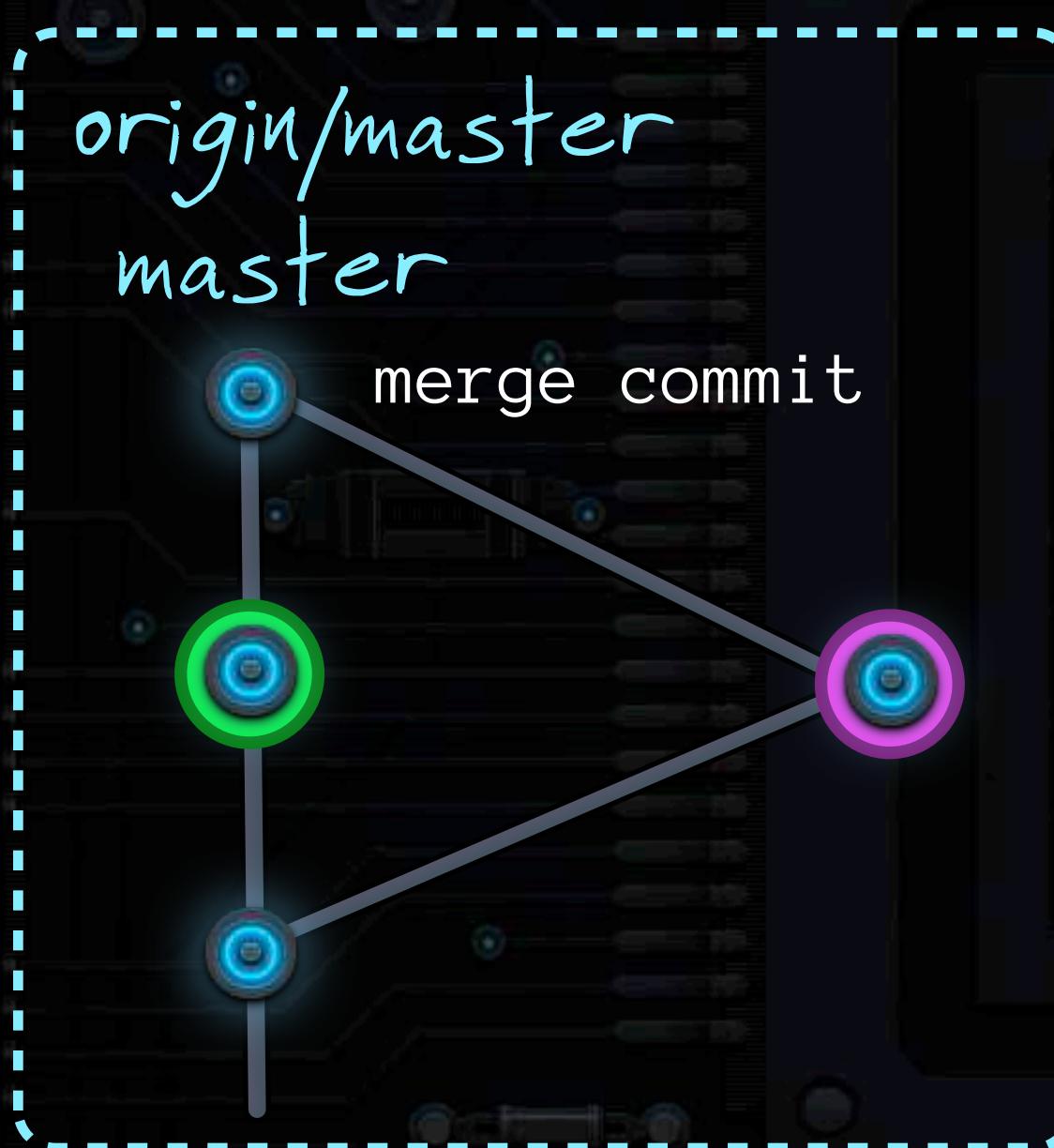
## git-real /

name	age	message
LICENSE	6 days ago	Add LICENSE and finish README. [Gregg]
README.txt	3 hours ago	Update Readme with lie. [Gregg]
product.rb	8 hours ago	Add product and store models. [Gregg]
shopping_cart.rb	3 hours ago	Add shopping cart [Gregg]

REBASE BELONG TO US

LEVEL 6

# MERGE COMMITS ARE BAD

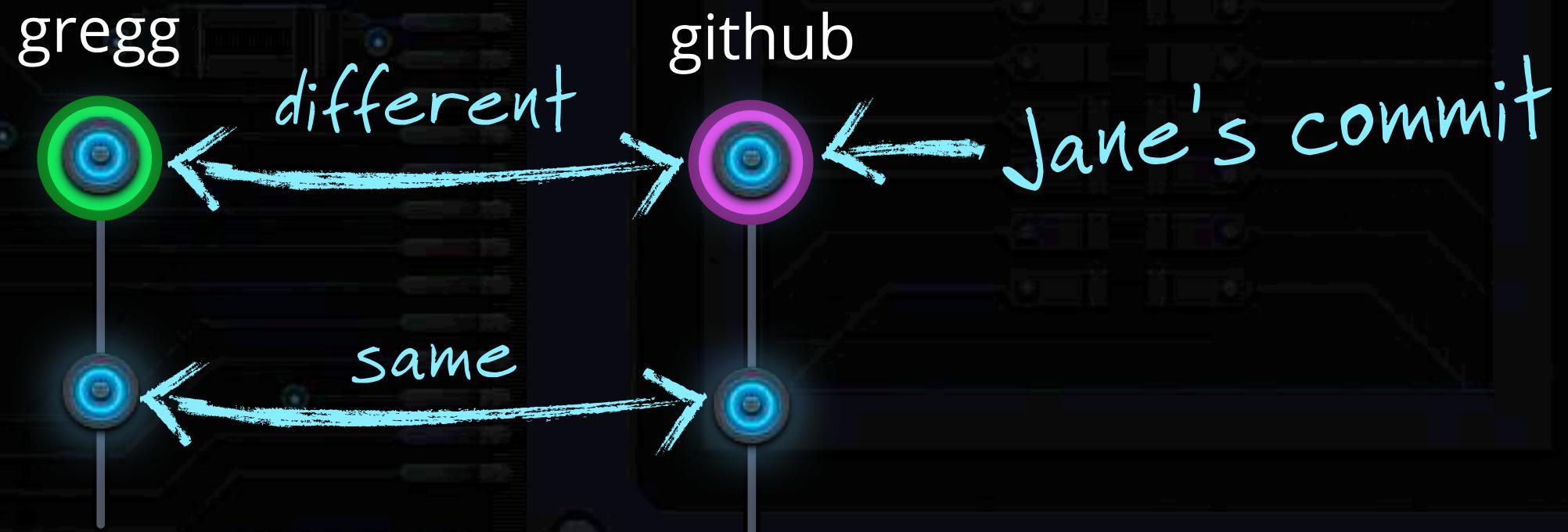


- Merge branch 'cats'
- Add Cats.
- Merge branch 'master' of http://...
- Update the Readme.
- Add product and store models.

Merge commits feel useless

# DIFFERENT GIT COMMITS

```
gregg $ git commit -am "Update the readme."  
[master c715339] Update the readme.  
1 file changed, 1 insertion(+), 1 deletion(-)
```



LEVEL 6 – REBASE BELONGS TO US

GIT  
REAL

# GIT PUSH REJECTED

```
gregg $ git push
```

```
To https://github.com/codeschool/git-real.git
```

```
! [rejected]          master -> master (non-fast-forward)
```

```
error: failed to push some refs to 'https://github.com/codeschool/git-real.git'
```

```
hint: Updates were rejected because the tip of your current branch is behind
```

```
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
```

```
hint: before pushing again.
```

```
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
$ git pull
```

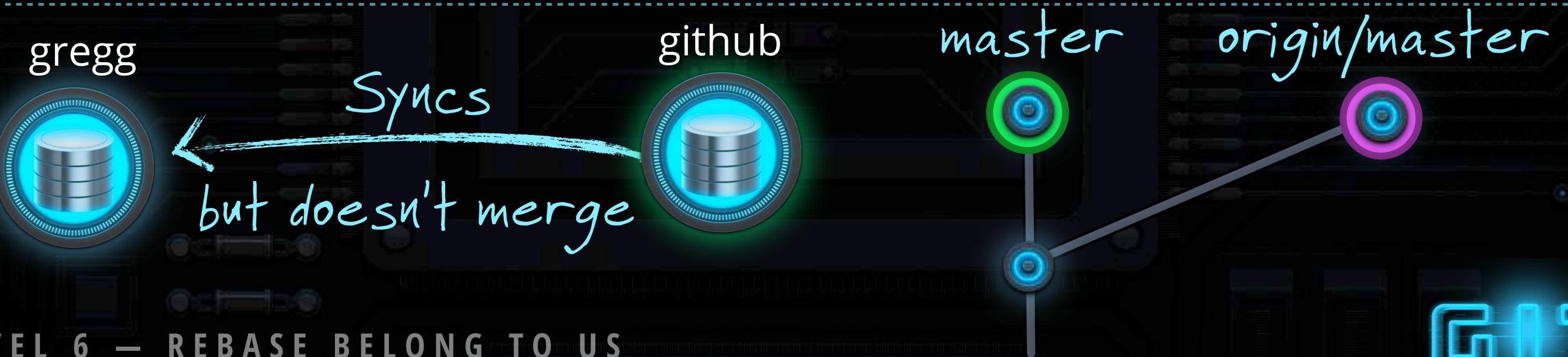
```
...
```

```
$ git push
```

Nope!

# FETCH

```
gregg $ git fetch  
remote: Counting objects: 5, done.  
remote: Compressing objects: 100% (2/2), done.  
remote: Total 4 (delta 0), reused 4 (delta 0)  
Unpacking objects: 100% (4/4), done.  
From https://github.com/codeschool/git-real  
      f35f2f1..71a4650  master      -> origin/master
```



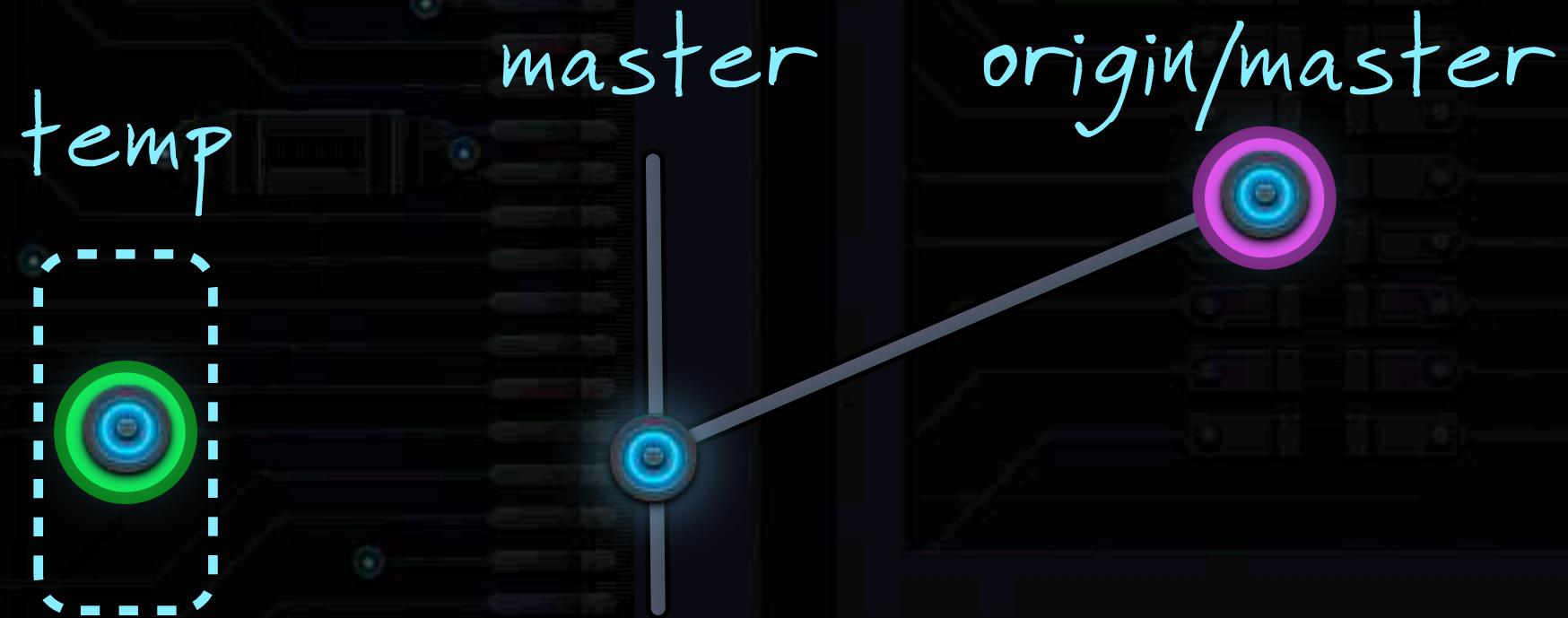
LEVEL 6 – REBASE BELONGS TO US

GIT  
REAL

# REBASE

```
gregg $ git rebase
```

1. Move all changes to master which are not in origin/master to a temporary area.



LEVEL 6 – REBASE BELONGS TO US

GIT  
REAL

# REBASE

```
gregg $ git rebase
```

1. Move all changes to master which are not in origin/master to a temporary area.
2. Run all origin/master commits.



3. Run all commits in the temporary area, one at a time.

# REBASE

```
gregg $ git rebase
```

1. Move all changes to master which are not in origin/master to a temporary area.
2. Run all origin/master commits.
3. Run all commits in the temporary area, one at a time.

master



Update the Readme.



Add product and store models.

No Merge Commit!

GIT  
REAL

# LOCAL BRANCH REBASE

```
$ git checkout admin  
Switched to branch 'admin'  
$ git rebase master  
...
```



# IF ALL GOES WELL, MERGE MASTER

```
$ git checkout master  
Switched to branch 'master'  
$ git merge admin
```

...

master

admin



# WHAT ABOUT CONFLICTS

gregg



Add shopping cart.



Add lie to readme.



same

github



Add truth to readme.



Add product and store models.

*Edited same file!*



LEVEL 6 – REBASE BELONGS TO US

GIT  
REAL

# FETCH

```
gregg $ git fetch
```

*master*



Add shopping cart.



Add lie to readme.



*origin/master*



Add truth to readme.



Add product and store models.

LEVEL 6 – REBASE BELONGS TO US

GIT  
REAL

# REBASE

```
gregg $ git rebase
```

1. Move all changes to master which are not in origin/master to a temporary area



origin/master



Add truth to readme.



Add product and store models.

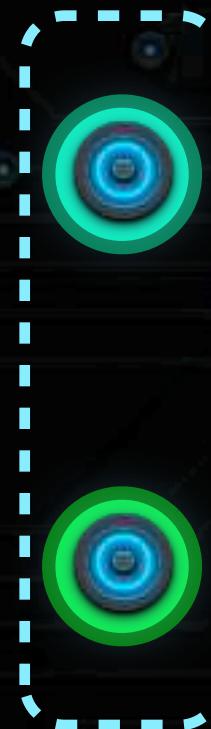
GIT  
REAL

# REBASE

```
gregg $ git rebase
```

2. Run all origin/master commits.

temp



master



Add truth to readme.

Add product and store models.

3. Run all commits in the temporary area, one at a time.

GIT  
REAL

# REBASE CONFLICT

```
gregg $ git rebase
```

First, rewinding head to replay your work on top of it...

Applying: Add lie to readme.

Using index info to reconstruct a base tree...

M README.txt

<stdin>:13: trailing whitespace.

the cake is a lie, and I am your father!

warning: 1 line adds whitespace errors.

Falling back to patching base and 3-way merge...

Auto-merging README.txt

CONFLICT (content): Merge conflict in README.txt

Failed to merge in the changes.

Patch failed at 0001 Add lie to readme.

master



When you have resolved this problem run "git rebase --continue".

If you would prefer to skip this patch, instead run "git rebase --skip".

To check out the original branch and stop rebasing run "git rebase --abort".



# REBASE CONFLICT

```
gregg $ git status
```

```
# Not currently on any branch.  
# Unmerged paths:  
#   (use "git reset HEAD <file>..." to unstage)  
#   (use "git add/rm <file>..." as appropriate to mark resolution)  
#  
# both modified: README.txt  
#  
no changes added to commit (use "git add" and/or "git commit -a")
```

Edit the README.txt

```
gregg $ git add README.txt
```

```
gregg $ git rebase --continue
```

Applying: Add lie to readme.

Applying: Add shopping cart

```
gregg $
```

master



# REBASED LOG

*master*



Add shopping cart.



Add lie to readme.



Add truth to readme.



Add product and store models.

GIT  
REAL

# HISTORY & CONFIGURATION

LEVEL 7

# VIEWING THE LOG

```
$ git log  
commit 915f242e262052b11c511dc07bef237fabb7ed85 ← SHA hash  
Author: Gregg <gregg@codeschool.com>  
Date:   Thu Jun 28 02:10:57 2012 -0700  
        Update index. ← commit message
```

# COLORIZING THE LOG

```
$ git config --global color.ui true  
$ git log  
commit 915f242e262052b11c511dc07bef237fabb7ed85  
Author: Gregg <gregg@codeschool.com>  
Date:   Thu Jun 28 02:10:57 2012 -0700  
        Update index
```

```
$ git log --pretty=oneline  
08f202691c67abd12eb886b587ac7b26d51005c7 Update index
```

# LOG FORMAT

```
$ git log --pretty=format:"%h %ad- %s [%an]"
```

placeholder	replaced with
%ad	author date
%an	author name
%h	SHA hash
%s	subject
%d	ref names

any string you want  
& placeholder data

run "git help log" for more options!

# PATCH

```
$ git log --oneline -p  
3ea7f70 I'm telling you, it's 'Octopi'.  
diff --git a/index.html b/index.html  
index 021a54e..640d66d 100644  
--- a/index.html  
+++ b/index.html  
@@ -8,7 +8,7 @@  
<nav>  
  <ul>  
    <li><a href="cat.html">Cats</a></li>  
-    <li><a href="octopus.html">Octopuses</a></li>  
+    <li><a href="octopi.html">Octopi</a></li>
```

# STATS

```
$ git log --oneline --stat
3ea7f70 I'm telling you, it's 'Octopi'.
 index.html | 2 +-  
 1 file changed, 1 insertion(+), 1 deletion(-)
96776a4 Add index.
 index.html | 30 ++++++-----  
 1 file changed, 15 insertions(+), 15 deletions(-)
```

# GRAPH

```
$ git log --oneline --graph
*   30b1f8f Merge branch 'bird' into master
| \
| * 8b8f950 Revise silly hipster name for bird aisle.
* | 915f242 Add emphasis.
| /
* 69728cd Update index descriptions.
```

visual representation of the branch merging into master

# DATE RANGES

```
$ git log --until=1.minute.ago
```

until

```
$ git log --since=1.day.ago
```

since (days)

```
$ git log --since=1.hour.ago
```

since (hours)

```
$ git log --since=1.month.ago --until=2.weeks.ago
```

since & until (relative)

```
$ git log --since=2000-01-01 --until=2012-12-21
```

since & until (absolute)

# DIFFS

```
$ git diff  
diff --git a/index.html b/index.html  
@@ -8,7 +8,10 @@  
<nav>  
  <ul>  
    <li><a href="cat.html">Cats</a></li>  
-    <li><a href="octopus.html">Octopuses</a></li>  
+    <li><a href="birds.html">Birds</a></li>  
+    <li><a href="hamsters.html">Hamsters</a></li>  
  </ul>  
</nav>  
</body>
```

*removed line* ↘

↑ *added lines*

# UNCOMMITTED CHANGES

```
$ git diff HEAD
diff --git a/index.html b/index.html
index 021a54e..1ceb9d6 100644
@@ -8,7 +8,10 @@
<ul>
<li><a href="cat.html">Cats</a></li>
...
diff --git a/octopus.html b/octopus.html
index 55806be..ce8a2c7 100644
@@ -2,6 +2,6 @@
<html lang="en">
...
```

*diff between last commit & current state*

*includes both staged and unstaged files*



# EARLIER COMMITS

```
$ git diff HEAD^
```

parent of latest commit

```
$ git diff HEAD^^
```

grandparent of latest commit

```
$ git diff HEAD~5
```

five commits ago

```
$ git diff HEAD^..HEAD
```

second most recent commit vs. most recent

# EARLIER COMMITS

```
$ git diff f5a6sdfsfsdfsdf9..4sdsdfsdfsdfb063f
```

*range of SHAs*

```
$ git log --oneline  
257256c cat  
4fb063f Add index  
f5a6ff9 Add catalog pages
```

```
$ git diff 4fb063f.. f5a6ff9
```

*range of abbreviated SHAs*

```
$ git diff master bird
```

*diff between two branches*

```
$ git diff --since=1.week.ago --until=1.minute.ago
```

*time-based diff*

# BLAME

```
$ git blame index.html --date short  
...  
96776a42 (Gregg 2012-06-29 9) <ul>  
96776a42 (Gregg 2012-06-29 10)   <li>Cats</li>  
3ea7f709 (Jane   2012-06-30 11)   <li>Octopi</li>  
96776a42 (Gregg 2012-06-29 12) </ul>
```

...  
commit hash author date line # content

# EXCLUDING FILES

```
$ git status  
# Untracked files:  
#   (use "git add <file>..." to include in what will be committed)  
#  
# experiments/ ← we don't want to commit this...
```

.git/info/exclude

```
experiments/      will exclude this folder from git
```

```
$ git status  
# On branch master  
nothing to commit (working directory clean)
```

the experiment directory is now invisible to git



# EXCLUDE PATTERNS

tutorial.mp4

exclude this file

\*.mp4

exclude all .mp4 files

experiments/

exclude directory

logs/\*.log

exclude .log files in logs directory

# EXCLUDING FROM ALL COPIES

```
$ git status  
# On branch master  
# Untracked files:  
#   (use "git add <file>..." to include in what will be committed)  
#  
# logs/server.log ← don't commit log files, they create conflicts
```

.gitignore

logs/\*.log

add this pattern

```
$ git status  
# On branch master  
nothing to commit (working directory clean)
```

no more logs

# MOVING FILES

```
$ git rm README.txt
```

```
$ git status  
# Changes to be committed:  
#  
# deleted: README.txt
```

DELETED from the local filesystem & untracked

```
$ git commit -m "Remove readme"
```

# UNTRACKING FILES

```
$ git rm --cached development.log
```

what if you're already tracking log files?

```
$ git status  
# Changes to be committed:  
#  
# deleted:    development.log
```

not deleted from the local file system, only from Git

# UNTRACKING FILES

.gitignore

logs/\*.log will ignore all .log files inside the logs directory

```
$ git add .gitignore
$ git commit -m "Ignore all log files."
[master bccdc8c] Ignore all log files.
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 .gitignore
 delete mode 100644 development.log
```

# CONFIG

```
$ git config --global user.name "Gregg Pollack"  
$ git config --global user.email "gregg@codeschool.com"
```

remember these? there's more

```
$ git config --global core.editor emacs use emacs for interactive commands
```

```
$ git config --global merge.tool opendiff use opendiff for merging conflicts
```

OS X only

# LOCAL CONFIG

```
$ git config user.email "spamme@example.com" sets email for current repo
```

```
$ git config --list
user.name=Gregg
user.email=gregg@codeschool.com
color.ui=true
core.editor=mate -w
user.email=spamme@example.com
```

same key can  
be set twice

```
$ git config user.email
spamme@example.com
```

the global config loaded first, then repo config

# ALIASES

aliases for log formats

```
$ git config --global alias.mylog \
"log --pretty=format:'%h %s [%an]' --graph"
```

```
$ git config --global alias.lol \
"log --graph --decorate --pretty=oneline --abbrev-commit --all"
```

```
$ git mylog
*   19f735c Merge branch 'admin' [Jane]
| \
| * 7980856 Add user admin [Jane]
* | 5c9ed90 Add dashboard. [Jane]
| /
* ab48a3f Create quantum cat. [Jane]
```



# ALIASES

git config alias.<name> <command>

```
$ git config --global alias.st status
```

git st git status

```
$ git config --global alias.co checkout
```

git co git checkout

```
$ git config --global alias.br branch
```

git br git branch

```
$ git config --global alias.ci commit
```

git ci git commit

```
$ git st  
# On branch master  
nothing to commit (working directory clean)
```

