Versioning Your Code:

~~Git and Github™:~~

Code Management and Sharing for Researchers

using Git and Github (™)

Eliezer Kanal

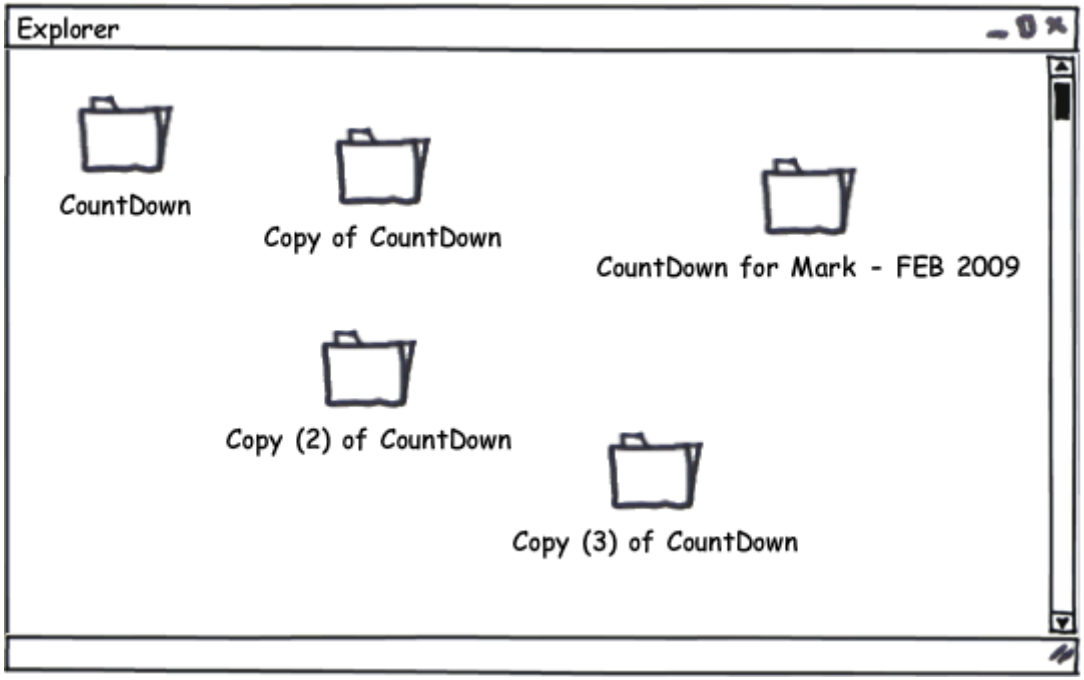
10/24/2011

Slides available online!

http://tinyurl.com/cnbcgit2

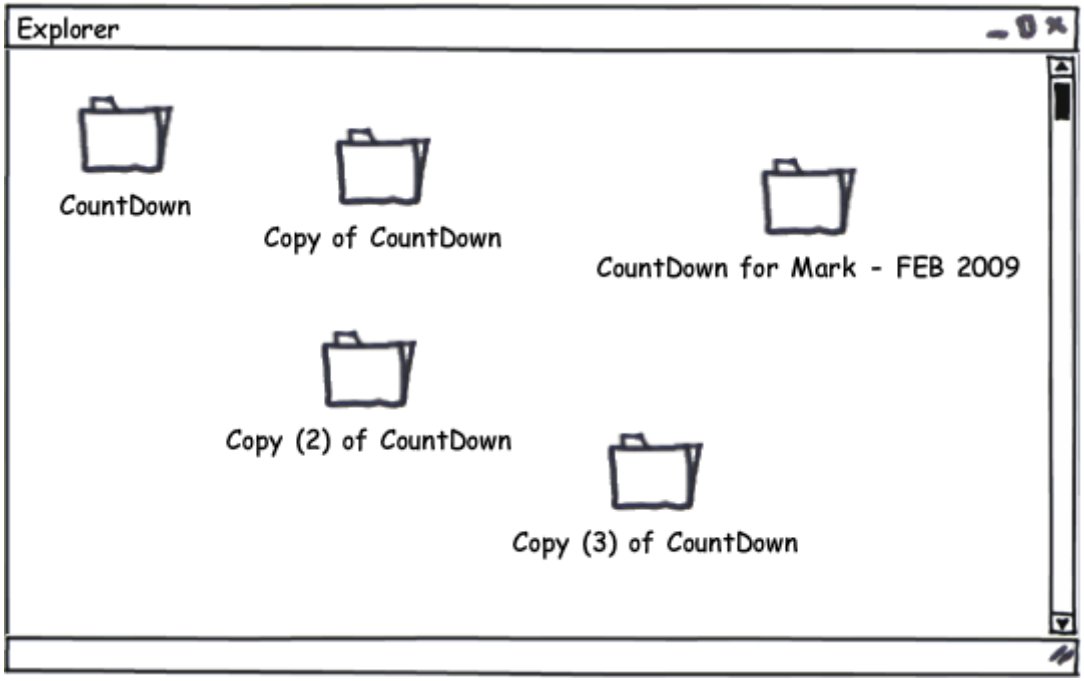
What is Versioning?

- Keeping track of changes to your code
- Documenting those changes
- You're probably doing it now...



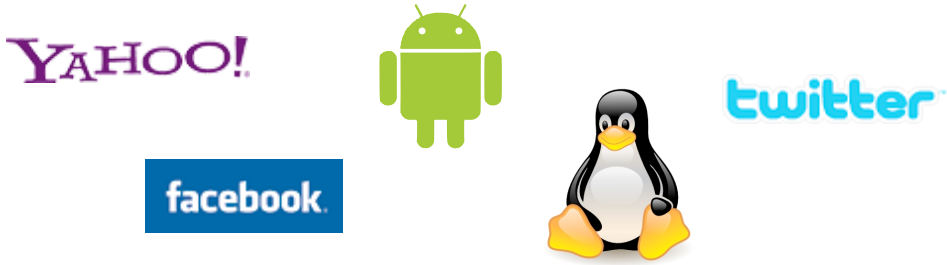
Why Change?

- Troubleshooting
- Logging
- Simplify adding features
- Simplify code sharing



What is Git?

- DVCS = Distributed Version Control System
- Technique for managing large/small coding projects

Large	
Small	Me, >1 million other people

What is Git?

- Two parts:

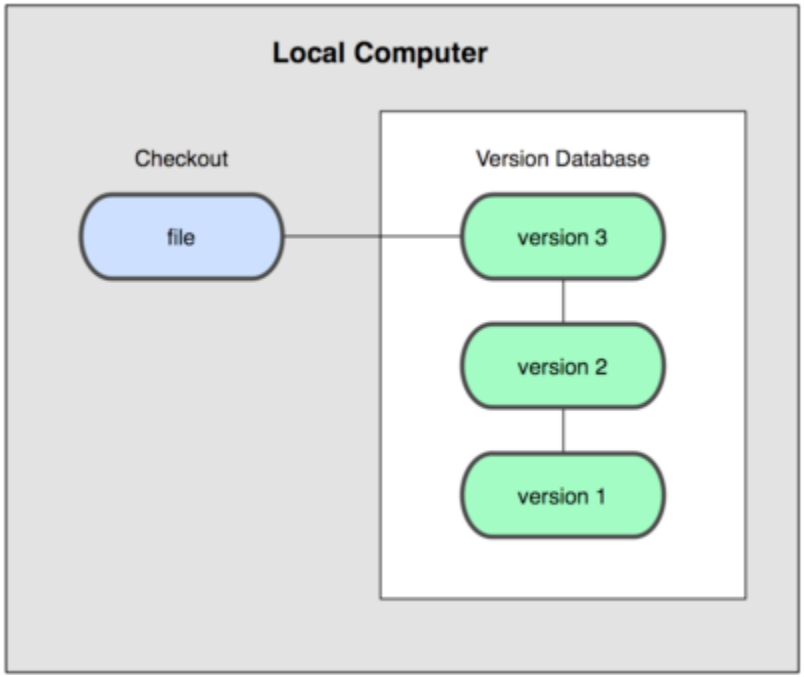
  1. “Database” (hidden folder) to hold project history

  2. Set of tools to interact with database

- Used via Command Line or GUI interfaces

What does Git do?

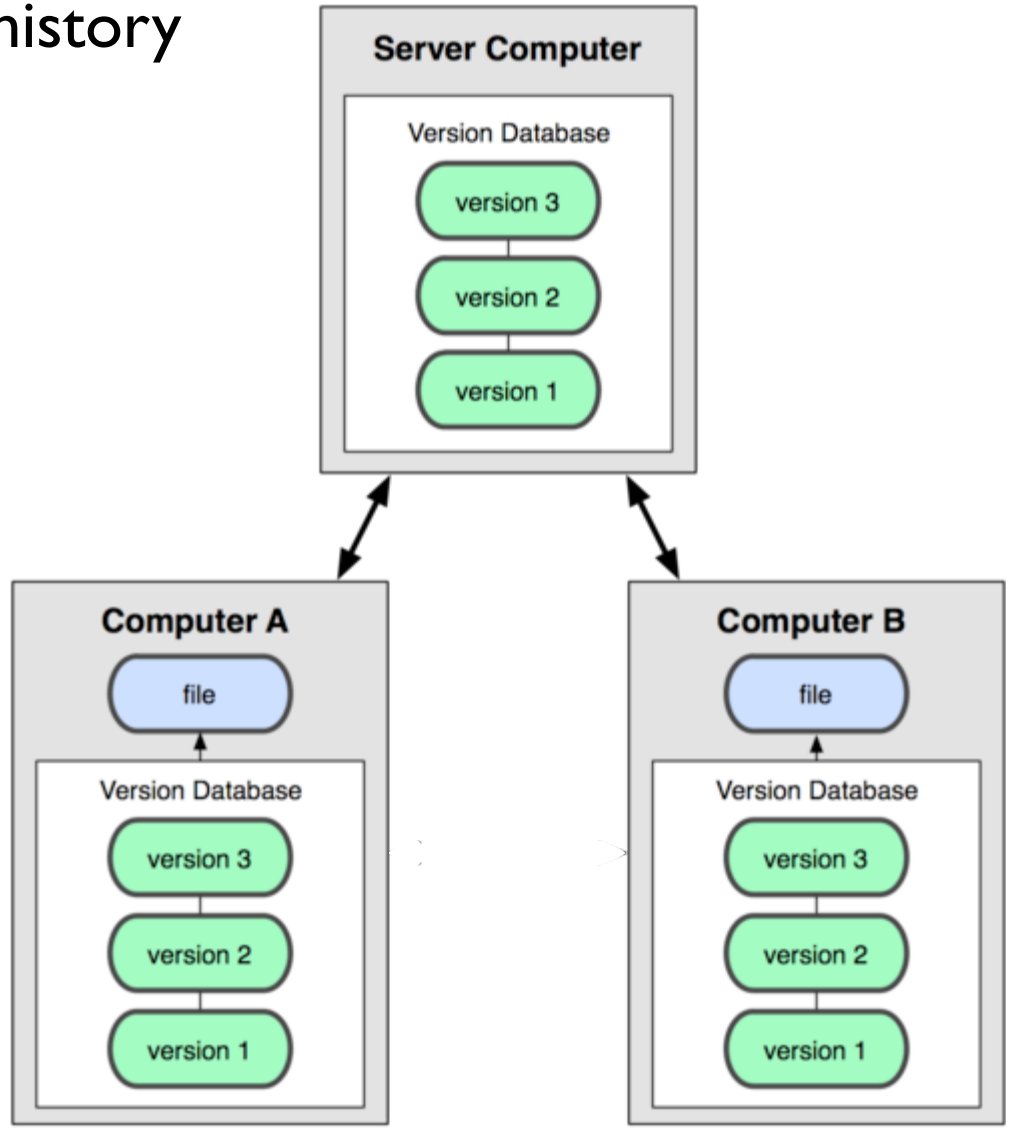
- Retain file history



What does Git do?

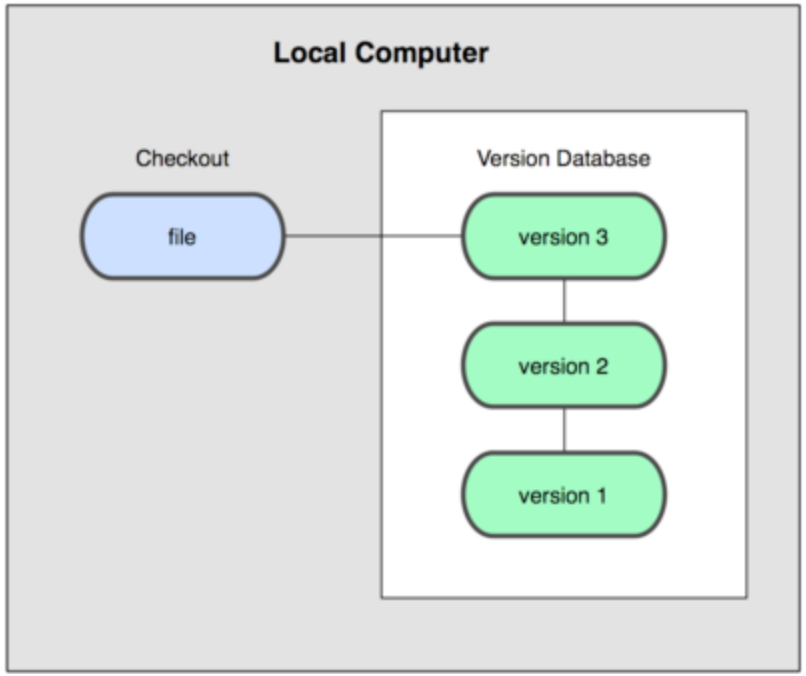
- Retain file history

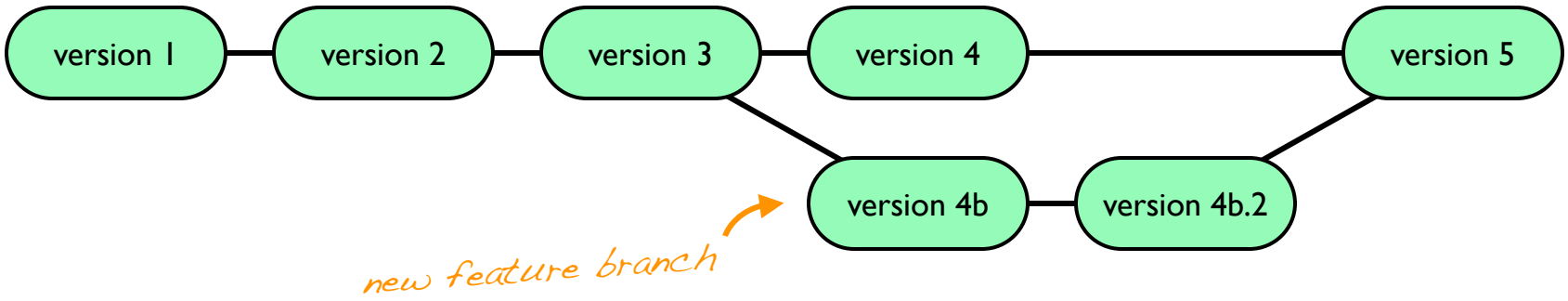
distributed



What does Git do?

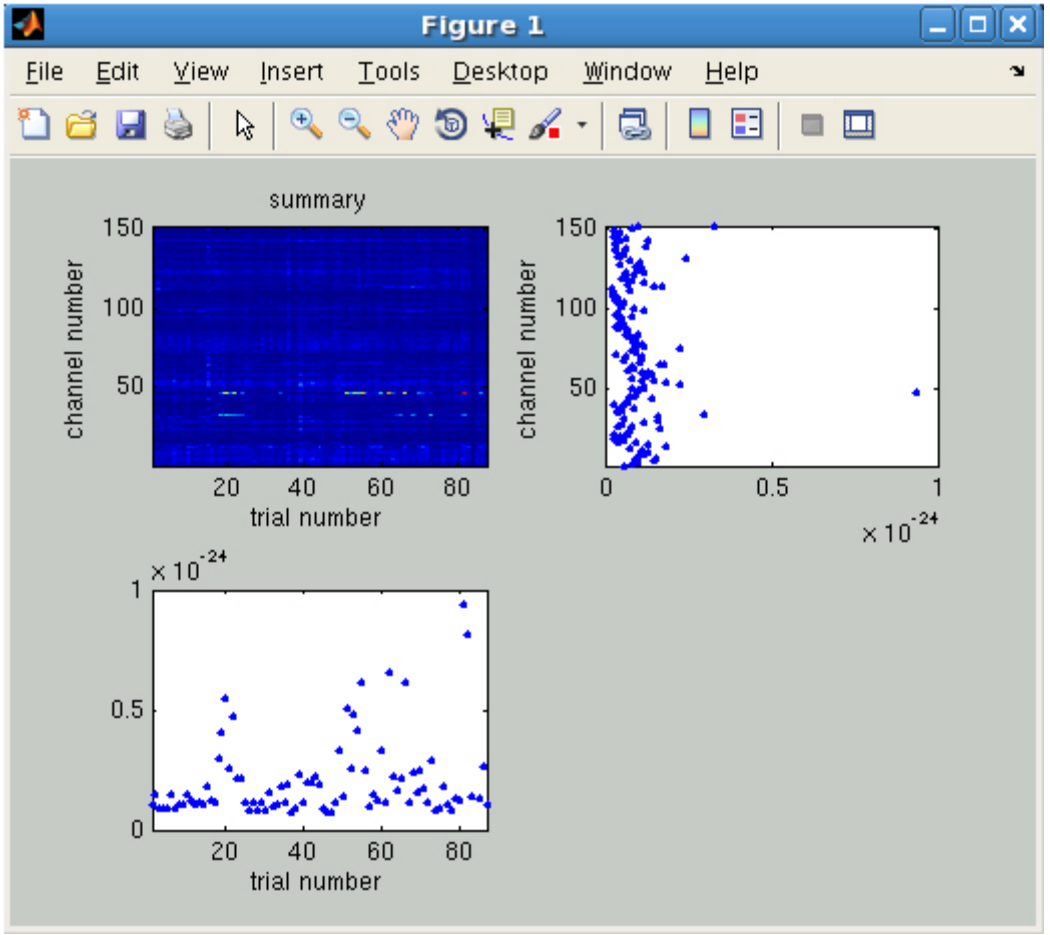
- Branching & Merging





Case Study – Analysis toolkit

- Improve interface, ~2 days work
  - …while analyzing data
- Branch!
  - Improvements on "development" branch
  - Work on "main" branch
  - Merge when feature completed

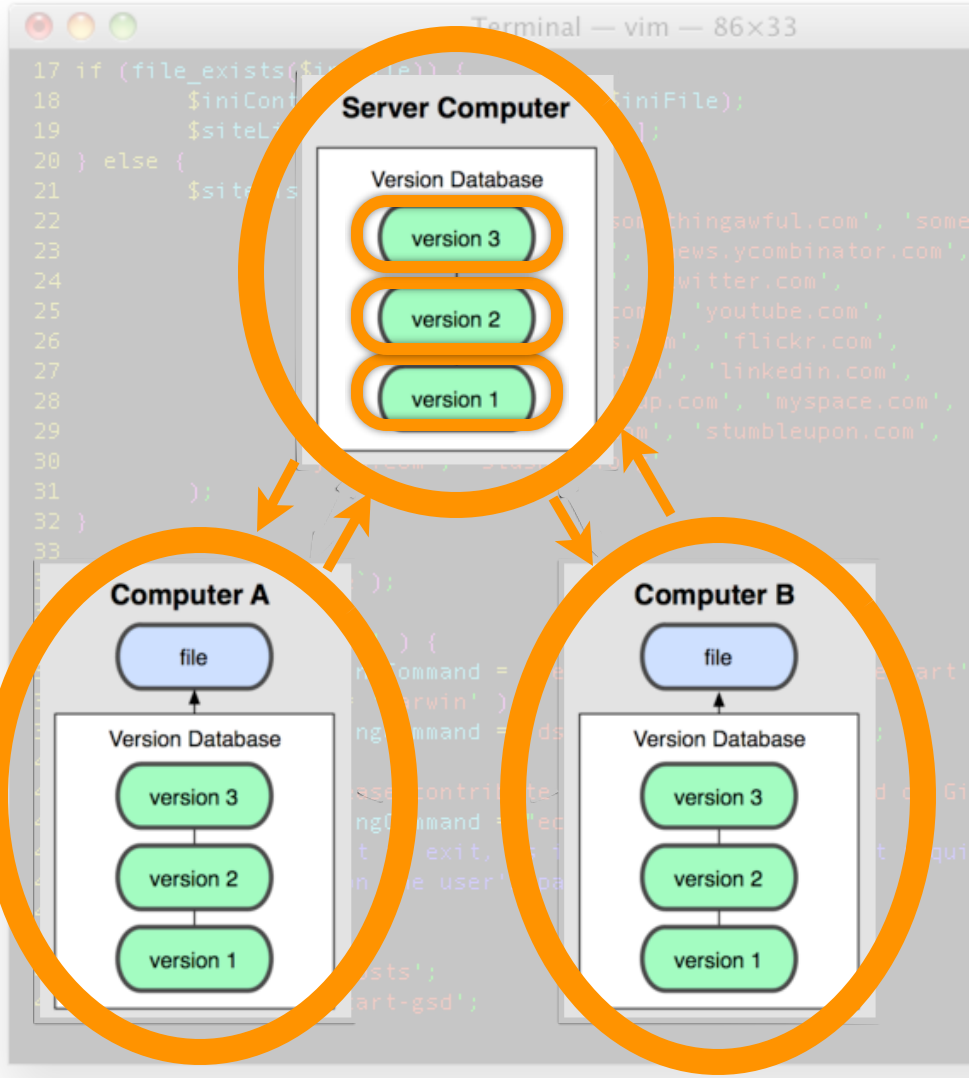


Case Study – Productivity tool

- get-bad word!-done: productivity tool, on github.com
- Downloaded, found bug
- Fixed bug, multiple local check-ins
- Sent change histories back via github.com

Case Study – Productivity tool

- `get-`~~bad word!~~`-done`: productivity tool, on `github.com`
- Downloaded, found bug
- Fixed bug, multiple local check-ins
- Sent change histories back via `github.com`



How Git* can help YOU

- Personally:
  - Keep track of changes
  - Fix bugs quicker
  - Share code easier
  - Simplify adding new features/changing code
- CNBC as a group:
  - "Don't reinvent the wheel"
  - More eyes → better code
  - Collaboration

* Or Mercurial, or Bazaar, or Monotone, or LibreSource, or...

This slide intentionally left blank

Git Usage

Key Concepts

Work is done here

Things "about to be stored" are put here

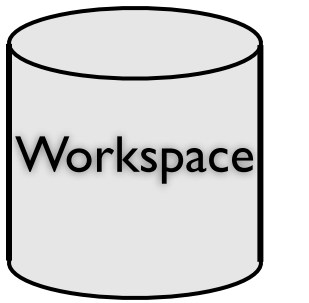
`$ git add`

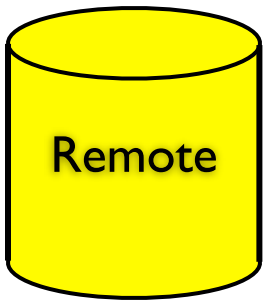
History, branches are stored here

`$ git commit`

Same as Repository, but remote

```
$ git push
$ git pull
```





- Create & edit some code

math.m

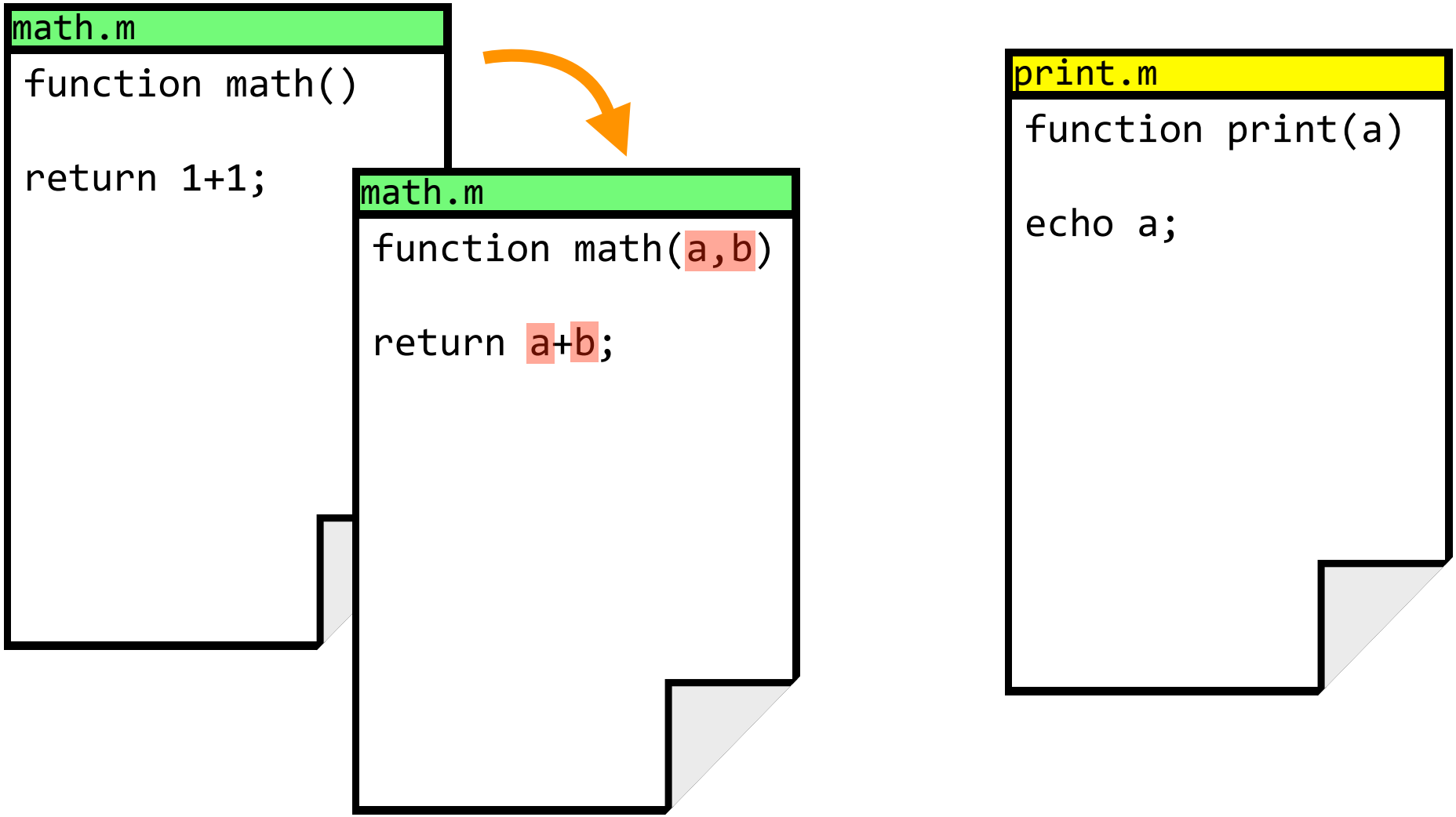
```
function math()

return 1+1;
```

- Move code to Staging Area

- Commit code

- Make changes, start working on new code



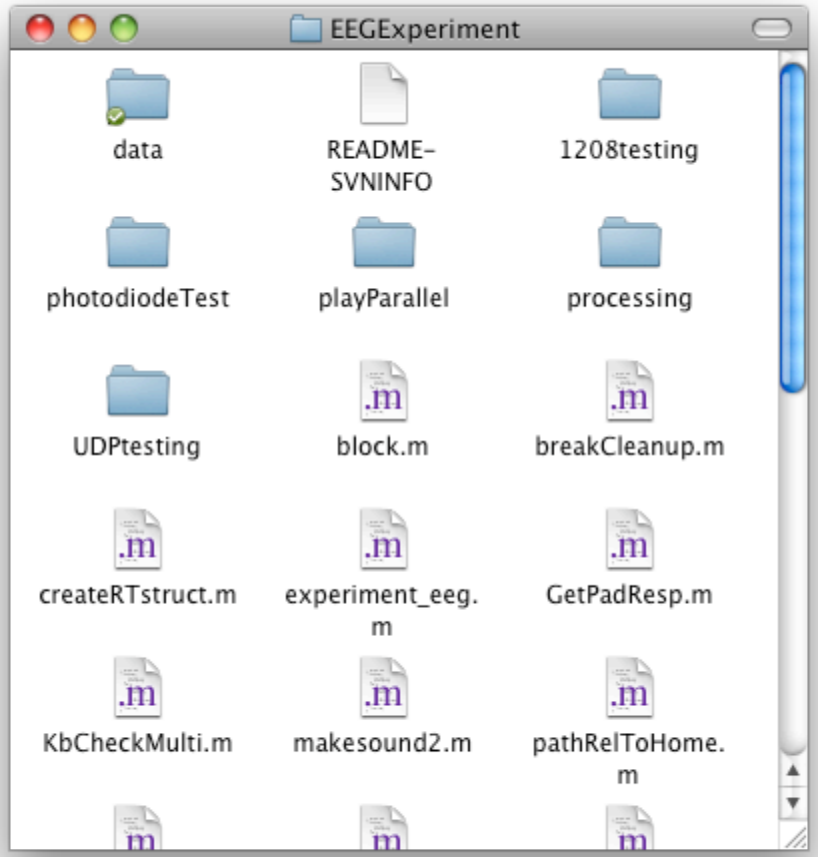
…wash, rinse, repeat:

- Move *completed* code to "Staging Area"
  - Leave the rest "unstaged"

- Commit code

Key Concepts

- Finder (Explorer) not suited for Git



- Are these in the workspace or staging area?
- What branch are we on?
- What revision is this?

Not enough information!

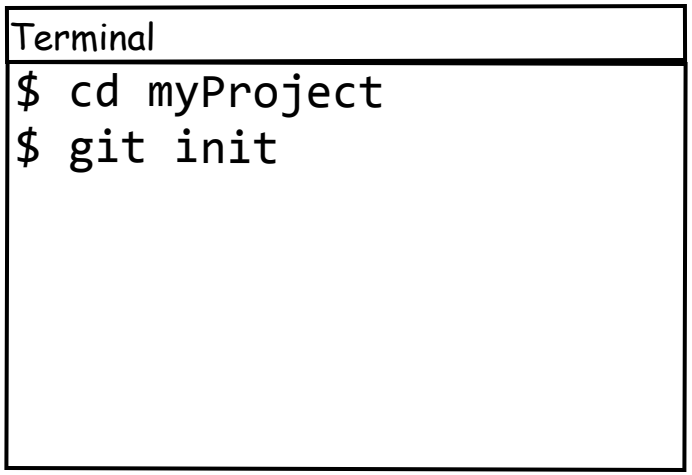
Git Interface

- Command line
- GUI
  - Github for Mac
  - GitExtensions
  - Others

All work slightly (significantly) differently

Example Usage

- Starting work on `myProject`:



That's it!

Command Line Interface

- Create & edit some code

math.m

```
function math()

return 1+1;
```

- Code is in working state; ready for “commit”

Terminal

```
$ git add math.m
$ git commit -m “created math function”
```

- Code is in working state; ready for “commit”

Terminal

```
$ git add math.m
$ git commit -m “created math function”
```

- Code is in working state; ready for “commit”

Terminal

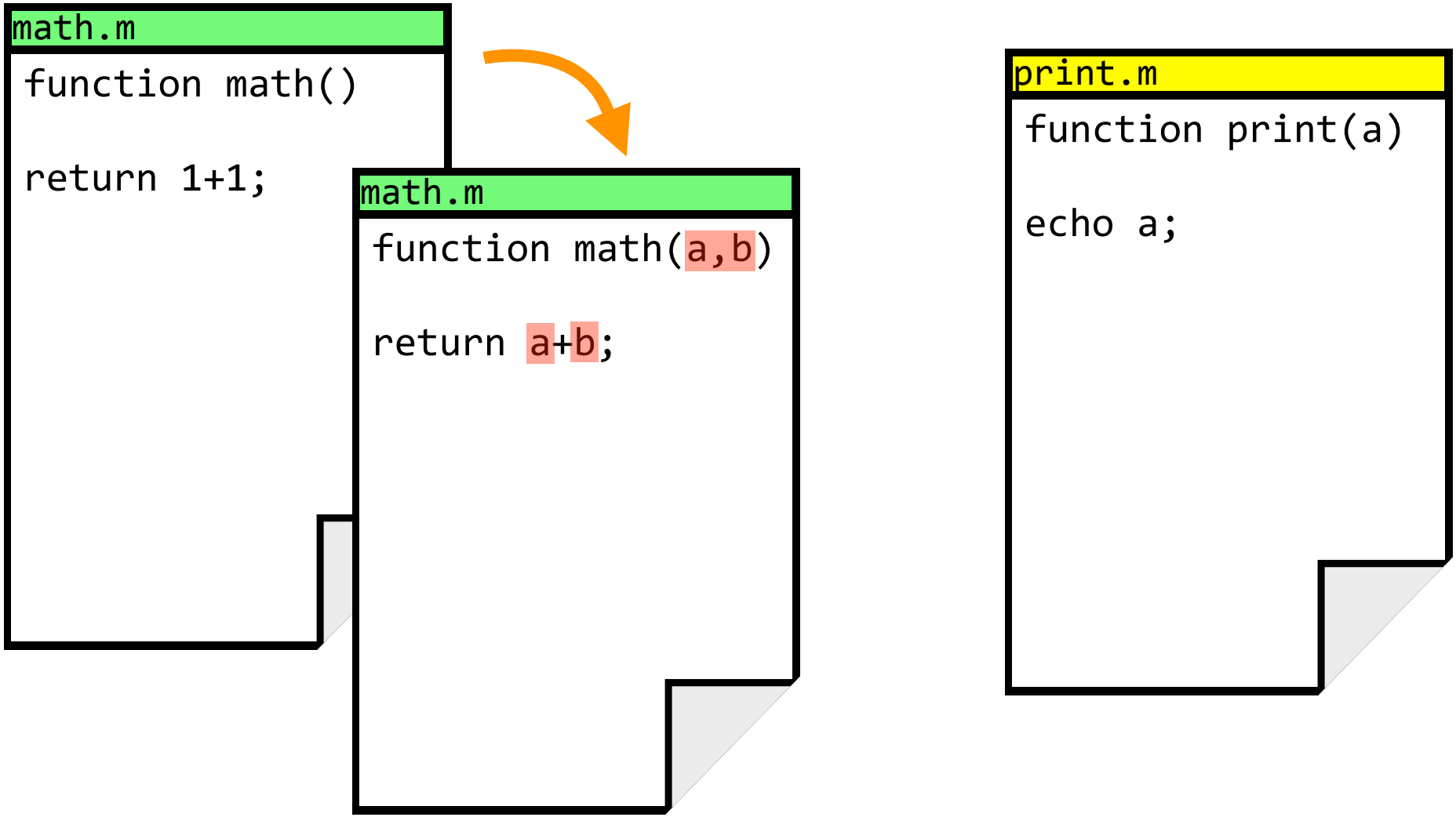
```
$ git add math.m
$ git commit -m “created math function”
```

- Code is in working state; ready for “commit”

Terminal

```
$ git add math.m
$ git commit -m “created math function”
```

- Make changes, add new code



- See what we've done

Terminal

```
$ git status
# On branch master
# Changes not staged for commit:
# (use "git add <file>..." to update what will be committed)
#
#   modified: math.m
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   print.m
no changes added to commit (use "git add" and/or "git commit -a")
```

- See what we've done

Terminal

```
$ git status
# On branch master
# Changes not staged for commit:
# (use "git add <file>..." to update what will be committed)
#
#   modified: math.m
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   print.m
no changes added to commit (use "git add" and/or "git commit -a")
```

- See what we've done

Terminal

```
$ git status
# On branch master
# Changes not staged for commit:
# (use "git add <file>..." to update what will be committed)
#
#   modified: math.m
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   print.m
no changes added to commit (use "git add" and/or "git commit -a")
```

- Add...

Terminal

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

- Add...

Terminal

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

- …and Commit

Terminal

```
$ git commit -m 'added variables to math, new print function'
[master a8ec60a] added variables to math, new print function
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 print.m
```

- …and Commit

Terminal

```
$ git commit -m 'added variables to math, new print function'
[master a8ec60a] added variables to math, new print function
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 print.m
```

- Let's see what we've done...

Terminal

```
$ git log
commit a8ec60a968fd64e1598a0601d85235be678fc8aa
Author: Eliezer Kanal <ekanal@cmu.edu>
Date:   Thu Jun 16 15:13:21 2011 -0400

    added variables to math, new print function

commit dbe573007aeb3e6bbb5ebfb3c7a17641d7290330
Author: Eliezer Kanal <ekanal@cmu.edu>
Date:   Thu Jun 16 15:11:04 2011 -0400

    created math function
```

Git Workflow – Complete

- Do some work
- See what you've changed, verify things look good:

```
$ git status
$ git difftool <filenames>
```

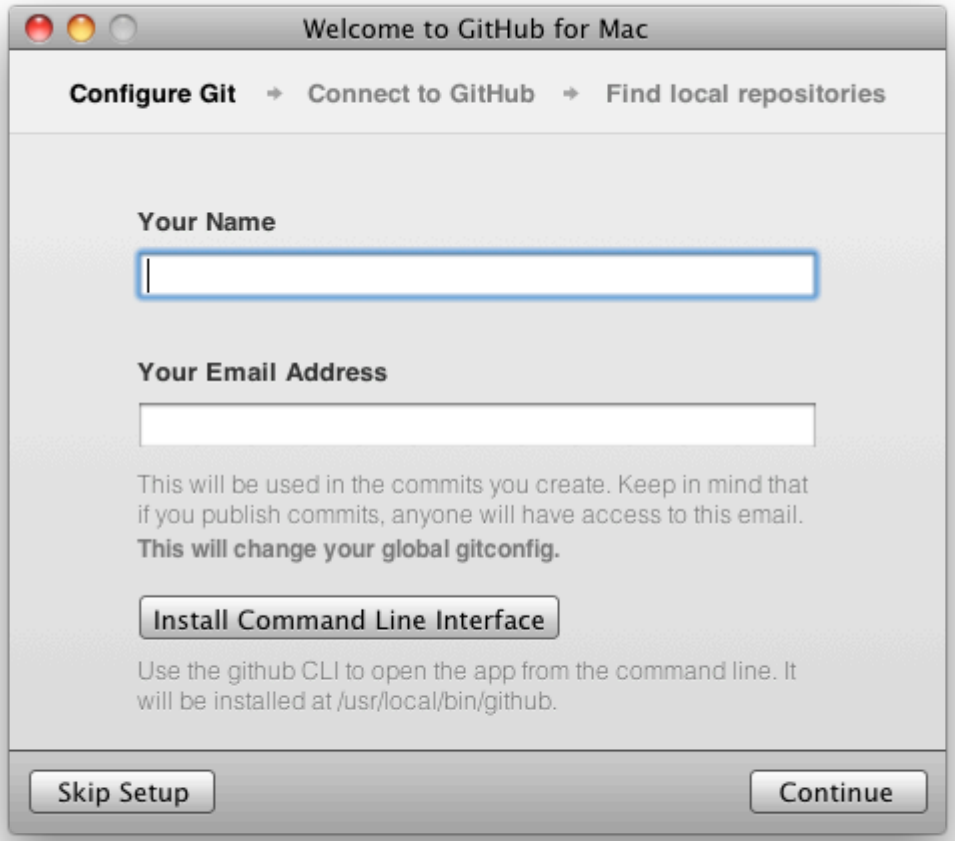
- Tell Git what you're ready to commit:

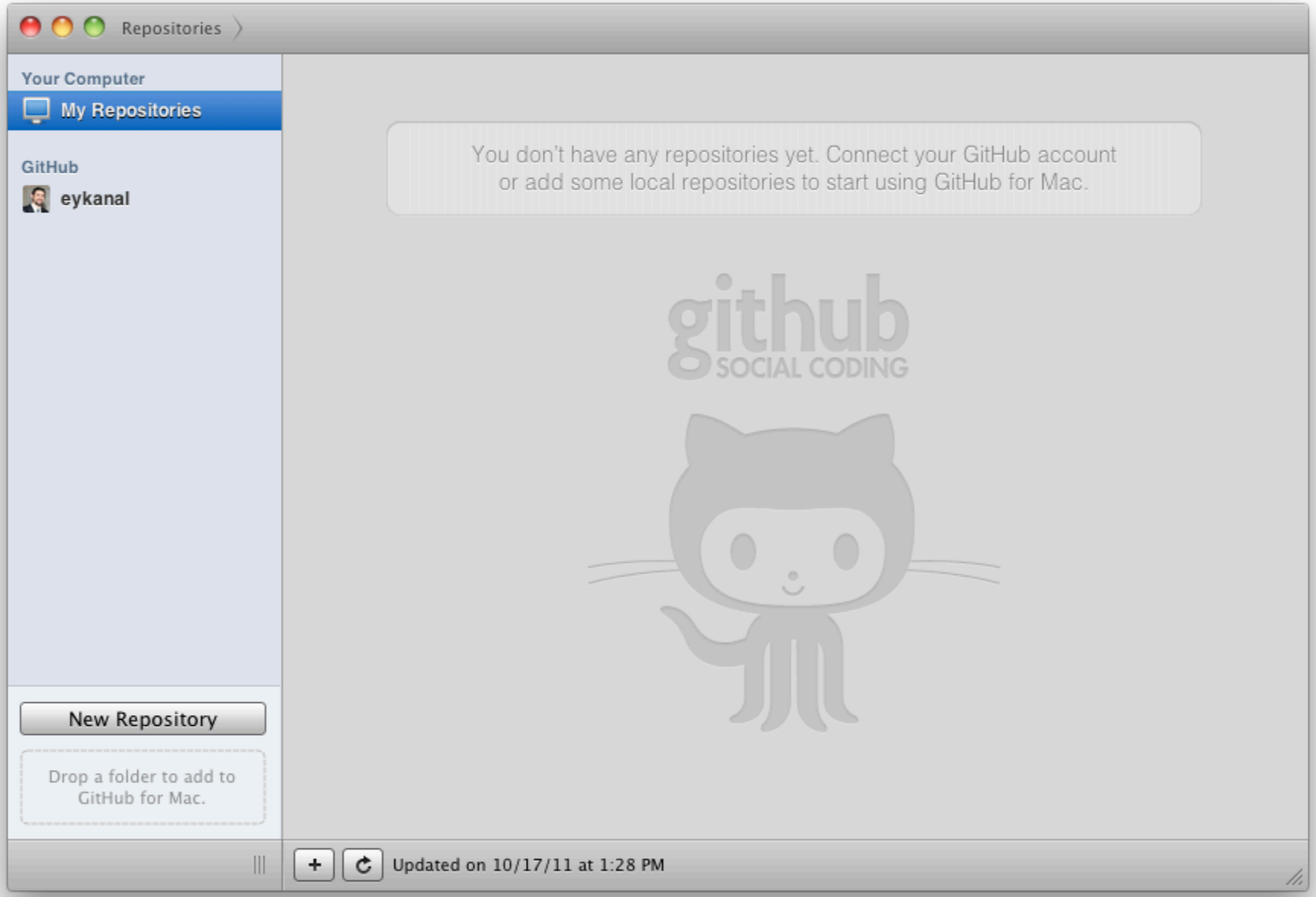
```
$ git add <filenames>
```

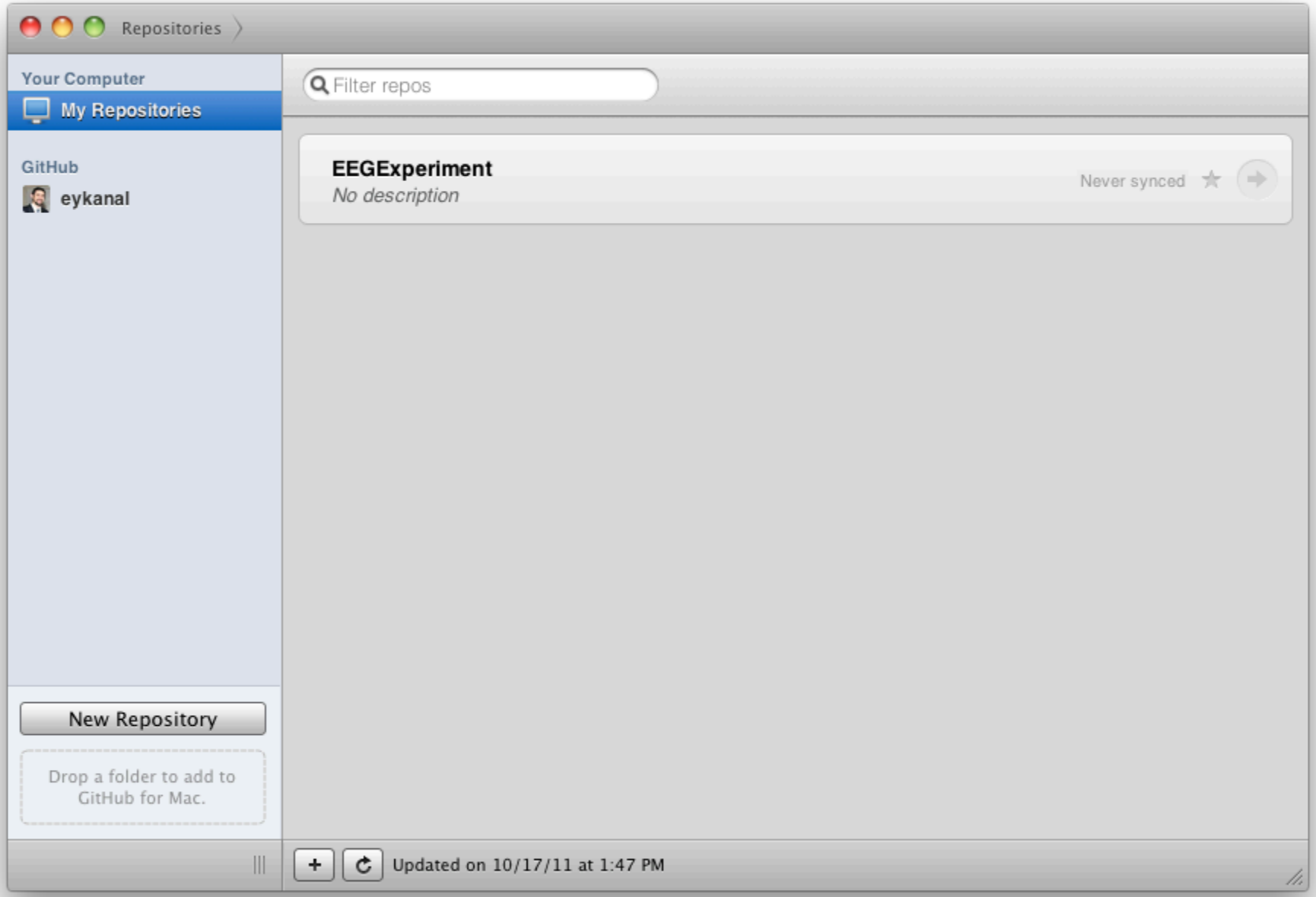
- Commit & log changes:

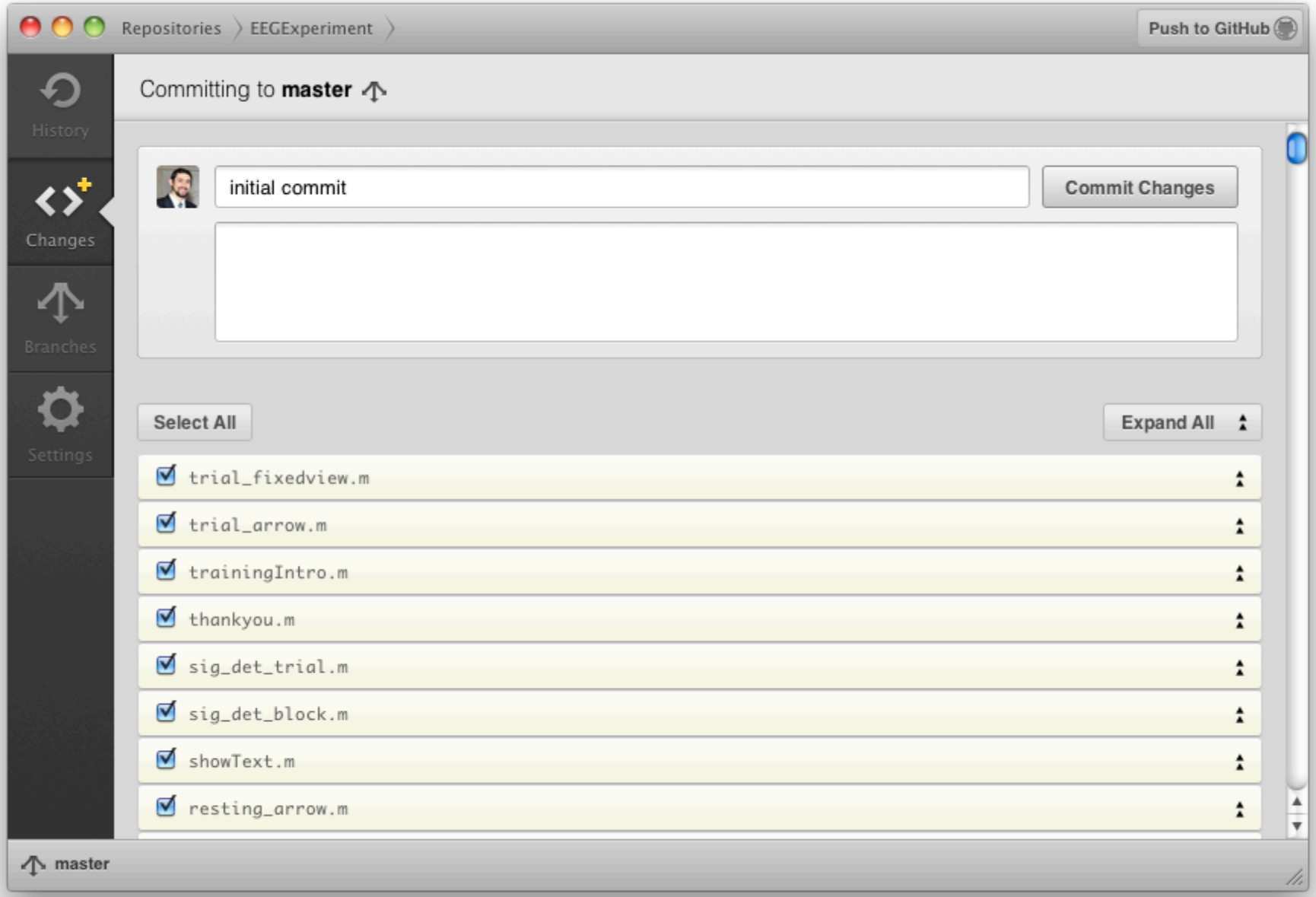
```
$ git commit <filenames>
(write detailed log notes)
```

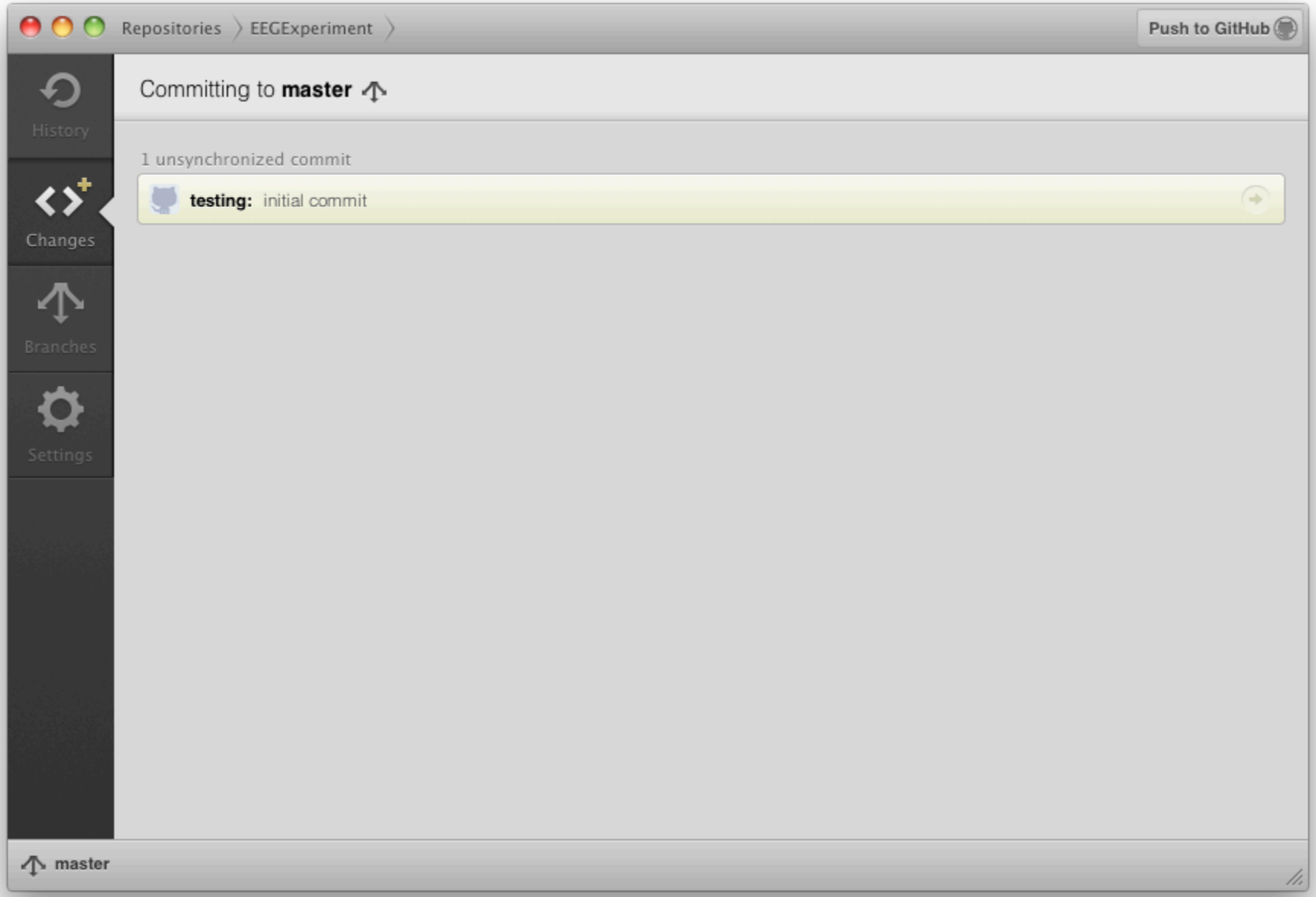
Github for Mac™

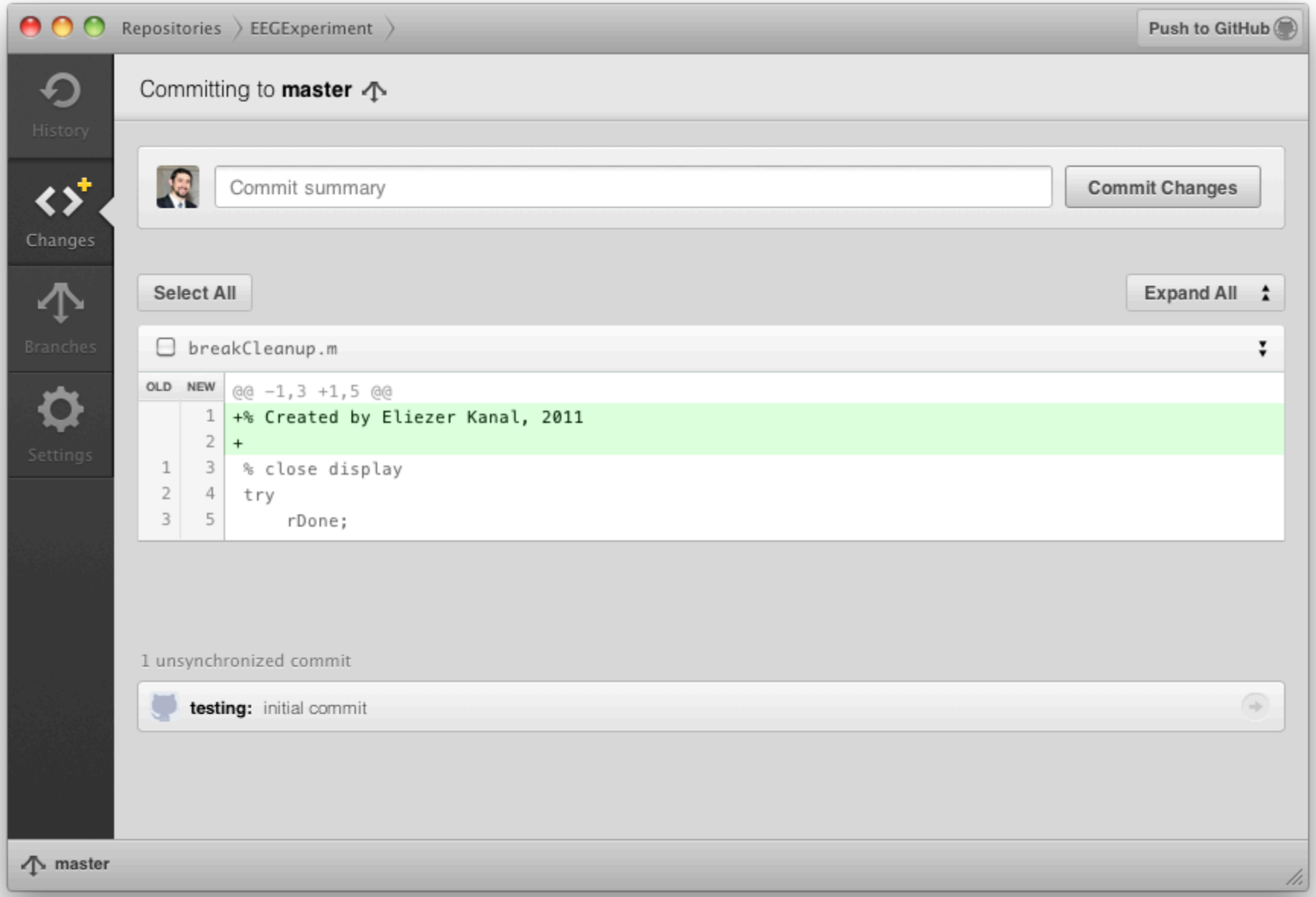

Welcome to GitHub for Mac
Configure Git → Connect to GitHub → Find local repositories
Your Name
Your Email Address
This will be used in the commits you create. Keep in mind that if you publish commits, anyone will have access to this email.
This will change your global gitconfig.
Install Command Line Interface
Use the github CLI to open the app from the command line. It will be installed at /usr/local/bin/github.
Skip Setup
Continue

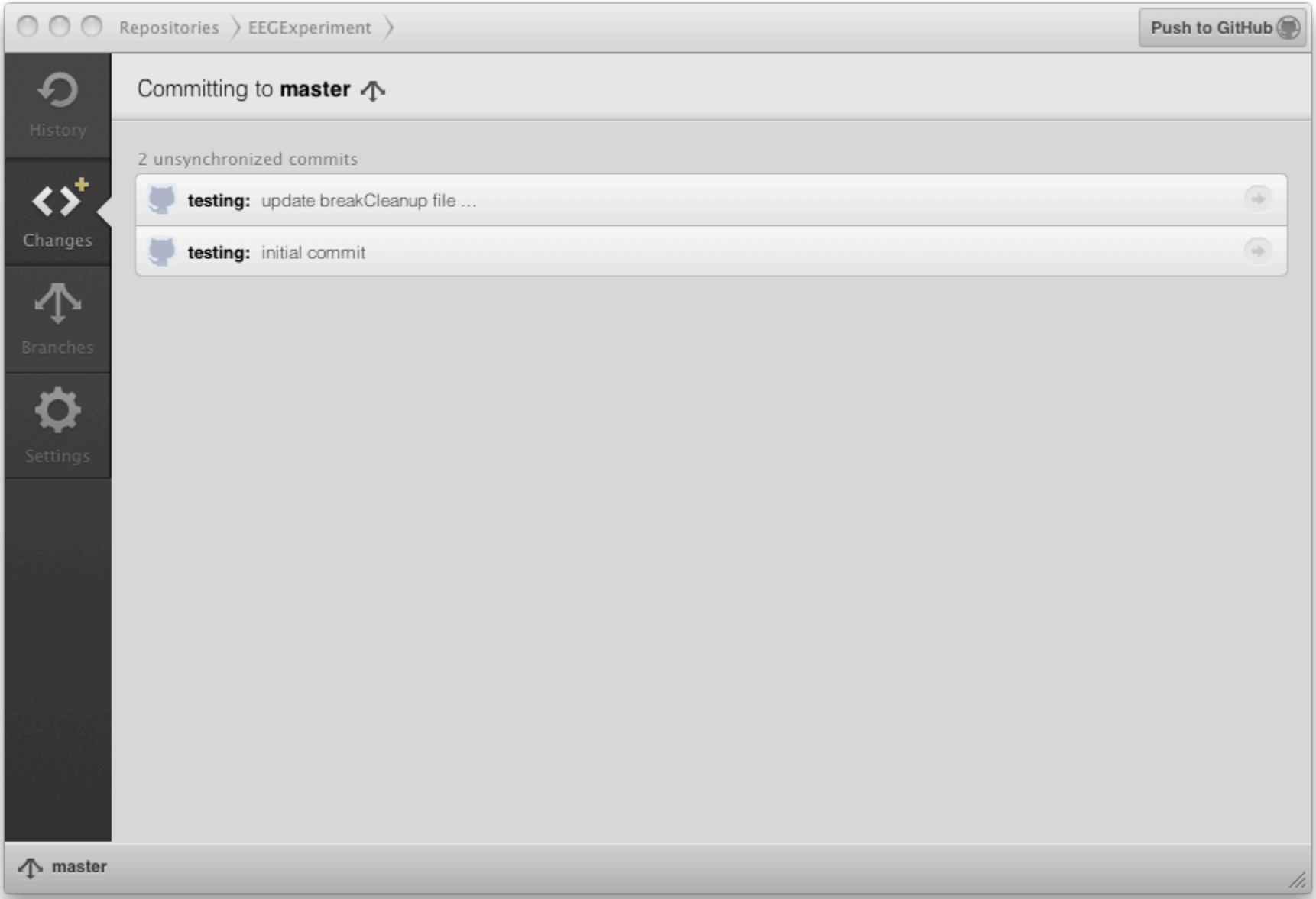

Repositories
Your Computer
My Repositories
GitHub
eykanal
You don't have any repositories yet. Connect your GitHub account or add some local repositories to start using GitHub for Mac.
github
SOCIAL CODING
New Repository
Drop a folder to add to GitHub for Mac.
Updated on 10/17/11 at 1:28 PM


Repositories
Your Computer
My Repositories
GitHub
eykanal
Filter repos
EEGExperiment
No description
Never synced
New Repository
Drop a folder to add to GitHub for Mac.
Updated on 10/17/11 at 1:47 PM


Repositories
EEGExperiment
Push to GitHub
History
Changes
Branches
Settings
Committing to master
initial commit
Commit Changes
Select All
Expand All
trial_fixedview.m
trial_arrow.m
trainingIntro.m
thankyou.m
sig_det_trial.m
sig_det_block.m
showText.m
resting_arrow.m
master


Repositories
EEGExperiment
Push to GitHub
History
Changes
Branches
Settings
Committing to master
1 unsynchronized commit
testing: initial commit
master


Repositories
EEGExperiment
Push to GitHub
History
Changes
Branches
Settings
Committing to master
Commit summary
Commit Changes
Select All
Expand All
breakCleanup.m
OLD
NEW
@@ -1,3 +1,5 @@
+% Created by Eliezer Kanal, 2011
+
% close display
try
rDone;
1 unsynchronized commit
testing: initial commit
master


Repositories
EEGExperiment
Push to GitHub
History
Changes
Branches
Settings
Committing to master
2 unsynchronized commits
testing: update breakCleanup file ...
testing: initial commit
master

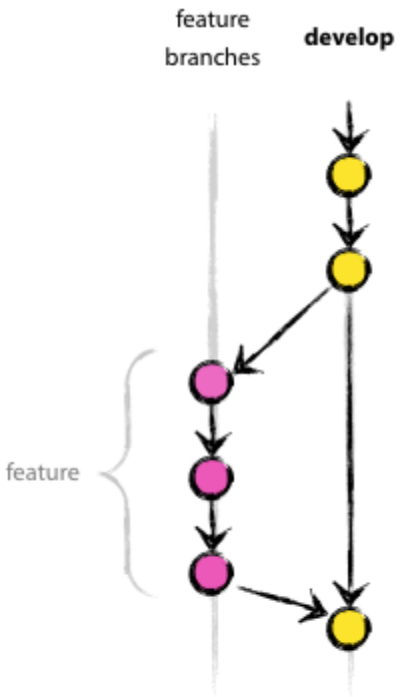
Branches

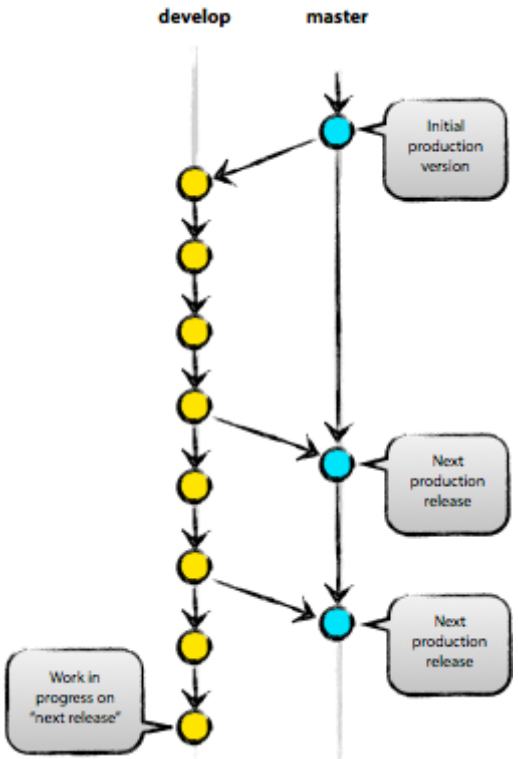
- *Question:* How can we integrate new features without messing up our current working code?

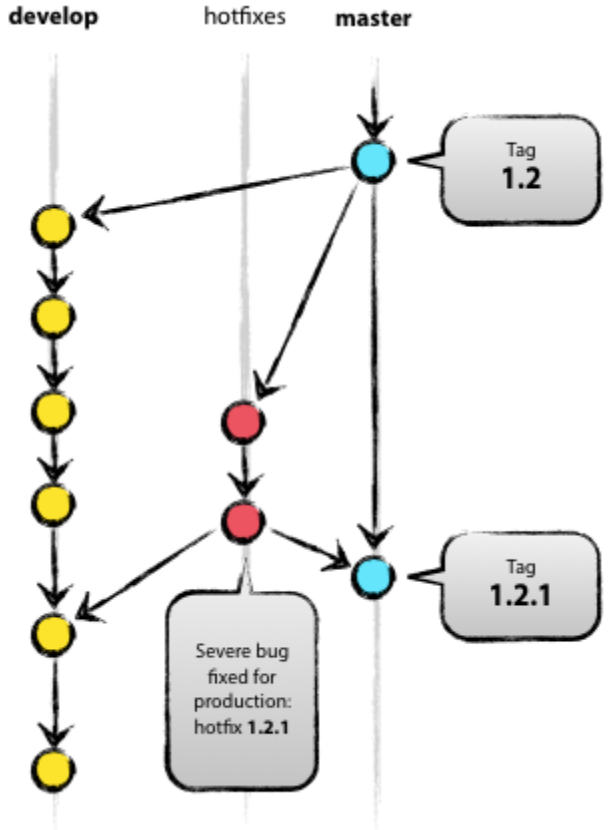
- Branches let us work in a "new directory" which we can later either delete or merge.

Branches & Tags

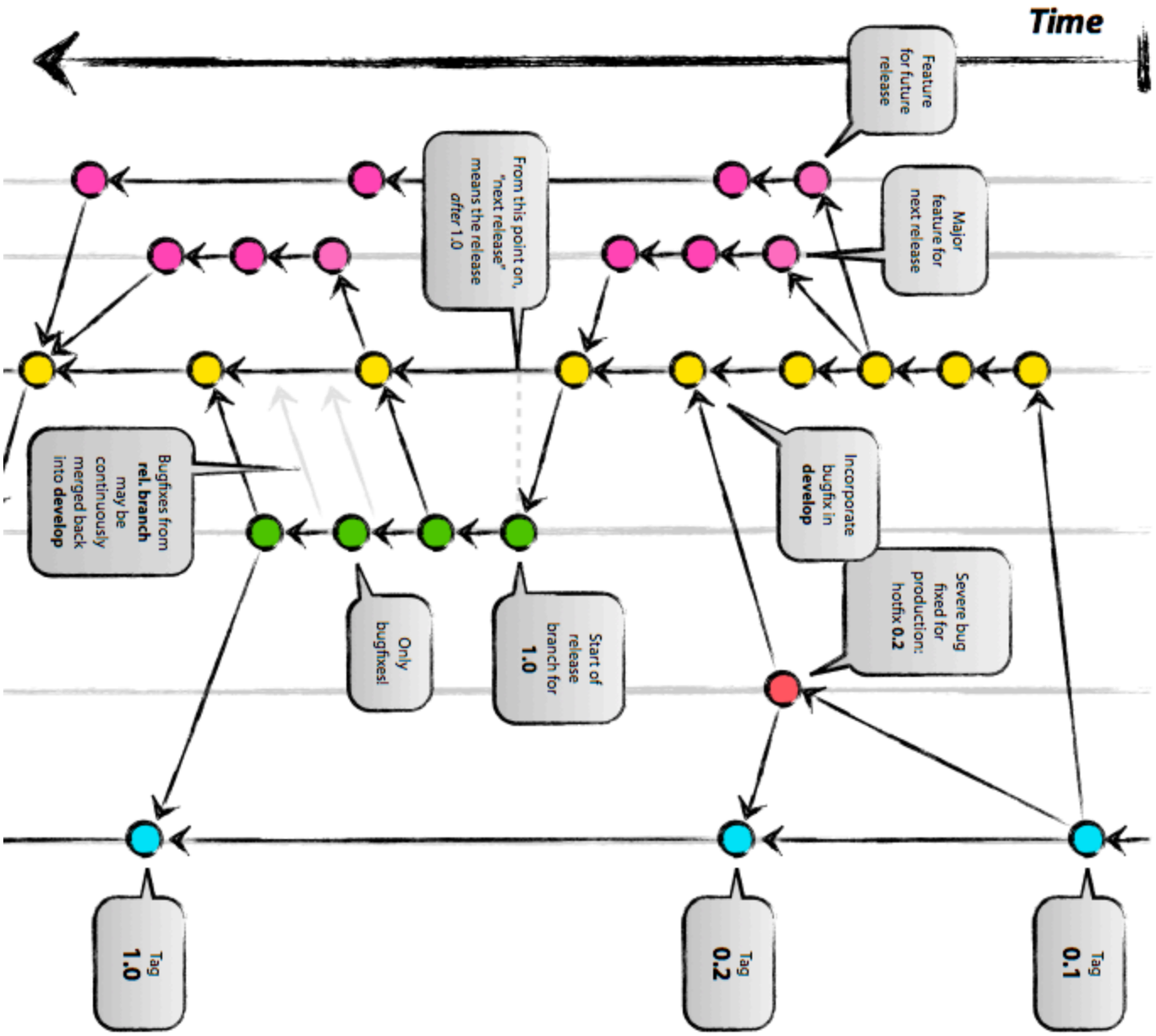
- Think of branches as separate folders, each with it's own content and history.







Branches – Go Nuts



Branches & Merges

- If a branch is a split in the graph, then merge is a merging of split branches
- One single command (SVN users rejoice!)
- Conflicts: Same code was changed in both branches
- Fix by hand, re-try the merge.
  - http://www.kernel.org/pub/software/scm/git/docs/user-manual.html#resolving-a-merge
  - better idea: Google "fixing a git merge conflict"

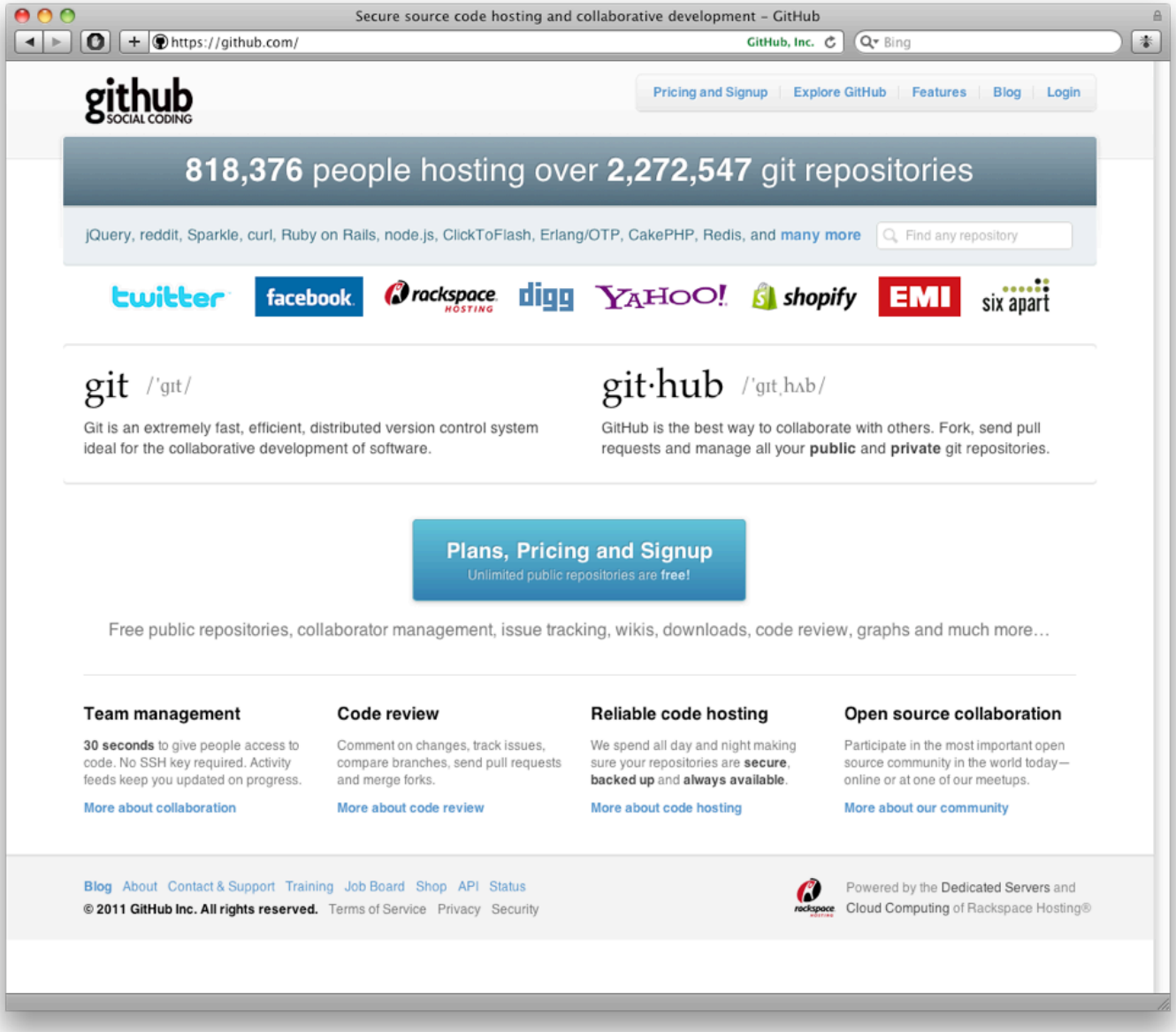
Branches & Merges

- Conflicts: Same code was changed in both branches
- Merge fails!
- Fix by hand, re-try the merge
  - http://www.kernel.org/pub/software/scm/git/docs/user-manual.html#resolving-a-merge
  - better idea: Search "fixing a git merge conflict"

Github.com

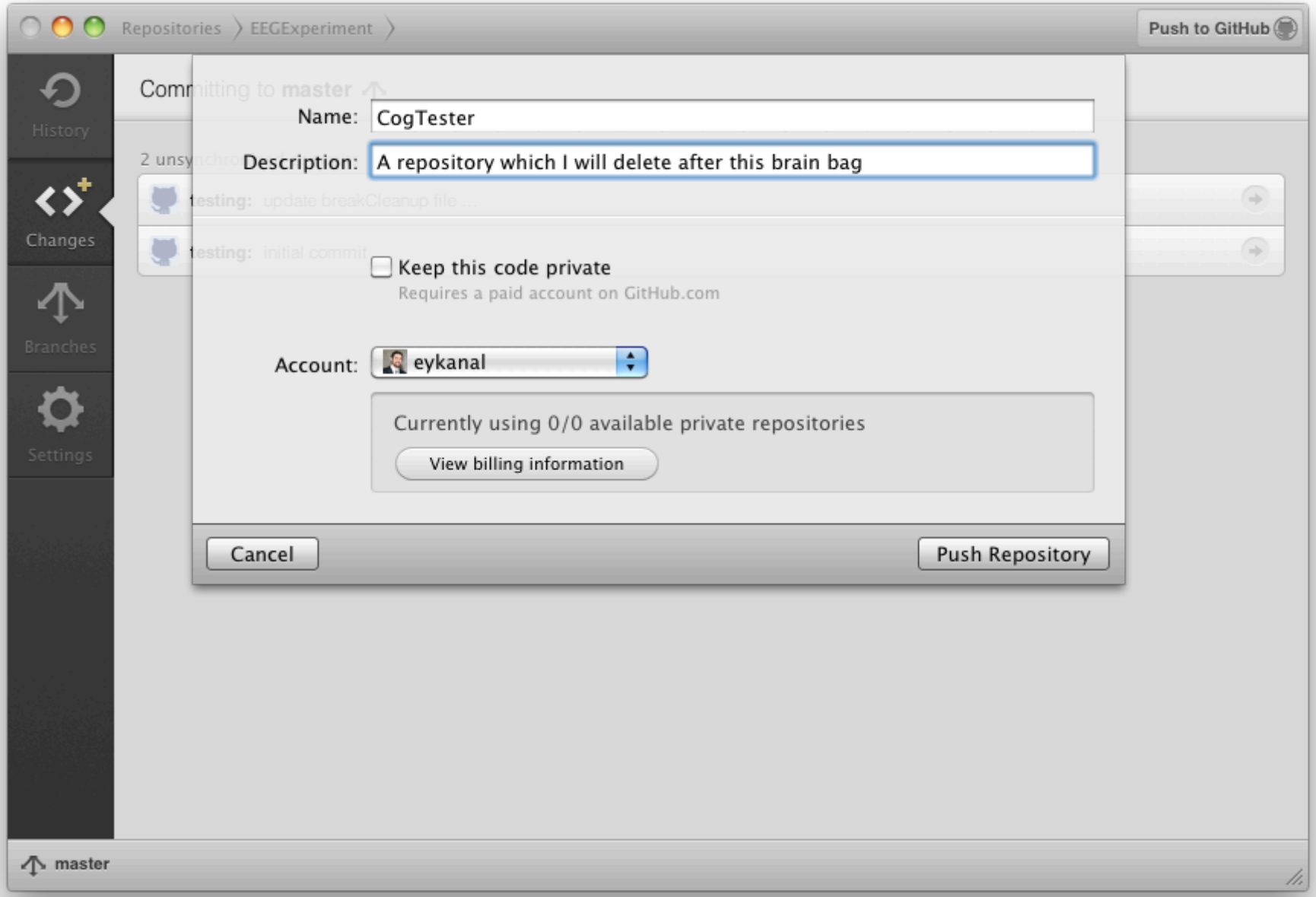
git·hub /ˈgɪtˌhʌb/

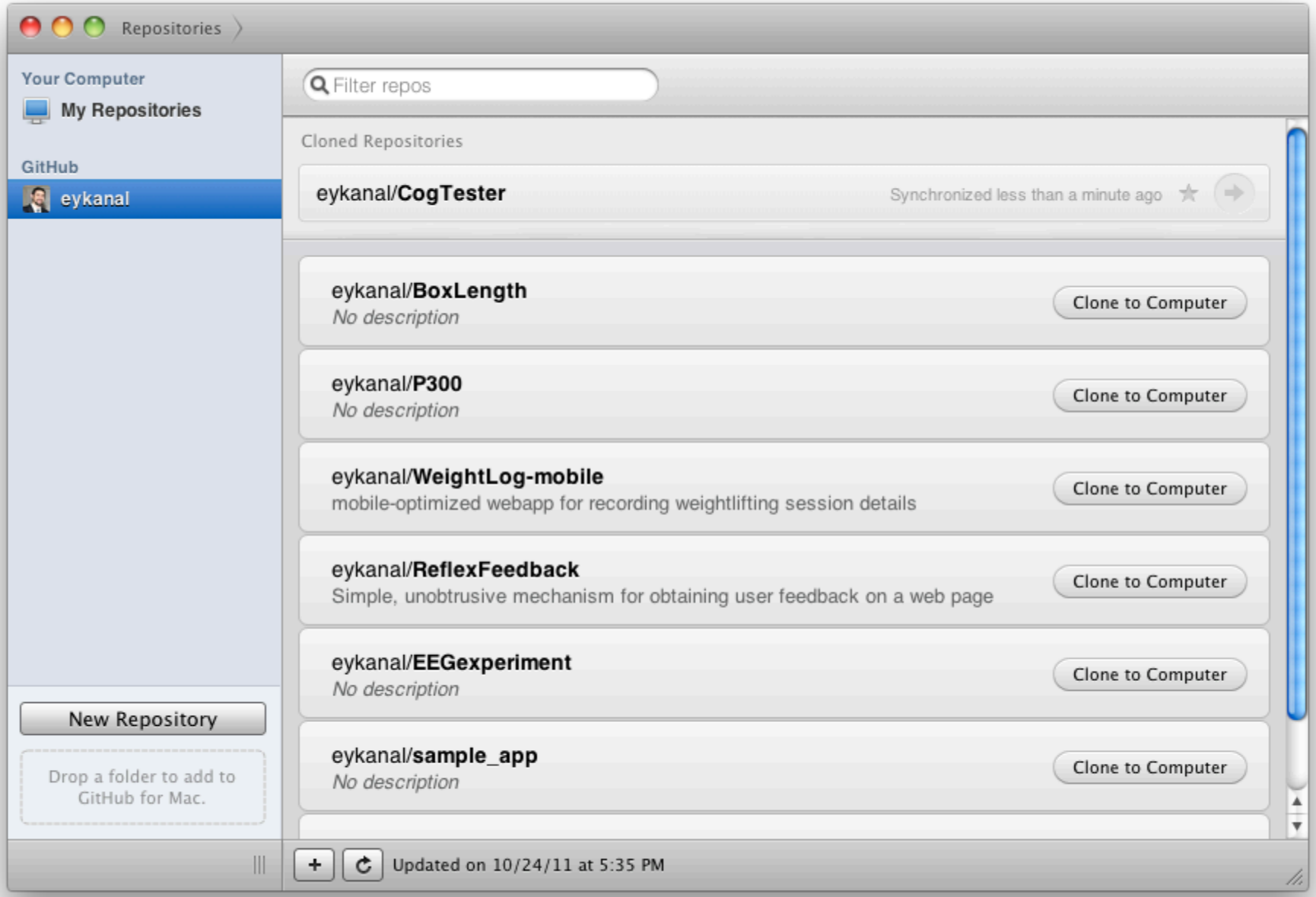
GitHub is the best way to collaborate with others. Fork, send pull requests and manage all your **public** and **private** git repositories.



Github.com

- Very easy collaboration
- Clear, detailed setup instructions
  - http://help.github.com/
- Easy "forking" for Github projects
  - Get your own copy of their code
  - Contribute back with "pull requests"

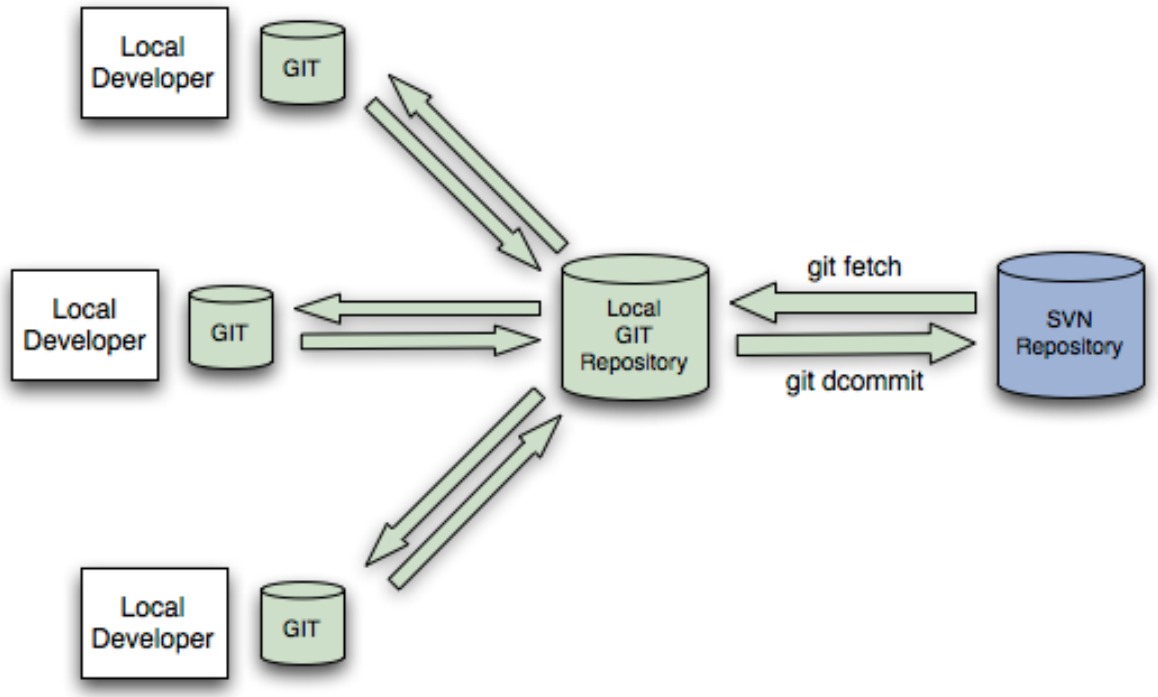

Repositories
EEGExperiment
Push to GitHub
History
Changes
Branches
Settings
Committing to master
Name:
CogTester
Description:
A repository which I will delete after this brain bag
Keep this code private
Requires a paid account on GitHub.com
Account:
eykanal
Currently using 0/0 available private repositories
View billing information
Cancel
Push Repository
master


Repositories
Your Computer
My Repositories
GitHub
eykanal
Filter repos
Cloned Repositories
eykanal/CogTester
Synchronized less than a minute ago
eykanal/BoxLength
No description
Clone to Computer
eykanal/P300
No description
Clone to Computer
eykanal/WeightLog-mobile
mobile-optimized webapp for recording weightlifting session details
Clone to Computer
eykanal/ReflexFeedback
Simple, unobtrusive mechanism for obtaining user feedback on a web page
Clone to Computer
eykanal/EEGexperiment
No description
Clone to Computer
eykanal/sample_app
No description
Clone to Computer
New Repository
Drop a folder to add to GitHub for Mac.
Updated on 10/24/11 at 5:35 PM

Demo

SVN integration

- Very easy to move from SVN → Git
- Works best as one-time deal, can work side-by-side
- http://orestis.gr/blog/2008/08/23/git-svn-tutorial/



Good Git Practice

- Check in often!
  - Rule of thumb… check in whenever code works
- Write detailed log notes! You'll thank yourself for it later!
- Branch for new features
  - Avoid working on the master branch, it should always be the "clean, working" code

References

- Pro Git - free online:
  http://progit.org/book/
- Branching:
  http://nvie.com/posts/a-successful-git-branching-model/
- Git tutorials and useful FAQ links:
  http://sixrevisions.com/resources/git-tutorials-beginners/
- Github:
  http://github.com/
- Git-SVN integration:
  http://orestis.gr/blog/2008/08/23/git-svn-tutorial/
- Google:
  http://google.com/
- Bing:
  http://bing.com/

Git Hands-On

Git Setup – one-tim

Recommend downloading xcode, just for the FileMerge program ("opendiff" when launched from command line)

- Identify yourself to Git:

```
$ git config --global user.name "Your Name"
$ git config --global user.email "your@email.com"
```

All your work is logged using the name/email you provide here.

- Give Git some color to make things easier to read:

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

- Line-ending bandaids:

```
$ git config --global core.safecrlf true
$ git config --global core.autocrlf input
```

Windows users: use "true" instead of "input"

Git Setup

- Begin using Git for a project:

```
$ cd projectFolder/
$ git init
Initialized empty Git repository in /projectFolder/.git/
```

  Creates the (hidden) `.git` folder. No files being tracked yet.

- Add files to repository:

```
$ git add .
$ git commit -m ‘first commit’
```

  Instructs Git to manage all files in the folder `/projectFolder/`.