



Michele Cox
Graduate Student
Maier Lab

Why Me?

*Likely **not** the most experienced Git user in either this room or in our department.*



Tools I'm glad I've learned
Make my work better

My Git Story in Emojis

2014



+ 6 months



- ❖ 1 Git project
 - Code Record
 - Backup

~**Value Added**~

Present



- ❖ **Use Git daily** on multiple projects
- ❖ **Manage** my own code/workflow
- ❖ **Collaborate** within lab
- ❖ **Contribute** to GitHub-hosted projects by other groups

My Git Story



Before Git


- **No record** of how my code had developed over time
- **No way to recover** files or systematically correct bugs
- **Depending heavily** on my own **memory** (it goes!)
- **No easy way to share** my work with others
- **Consumer (at best)** of other people's work

After Git

- **Automatic** (or close to it) **record** of my projects
- **Access to all past versions** of every piece of code
- **Save my memory** for more important things
- **System for sharing** and collaborating
- **Contributing member** of a larger development community

My Git Story

New Analysis Software



Cold Spring Harbor Laboratory

bioRxiv

beta

THE PREPRINT SERVER FOR BIOLOGY

HOME | AB

Search

New Results

Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels

Marius Pachitariu, Nicholas Steinmetz, Shabnam Kadir, Matteo Carandini, Kenneth D Harris
doi: <https://doi.org/10.1101/061481>

This article is a preprint and has not been peer-reviewed [what does this mean?].

Authors put it on GitHub



cortex-lab / KiloSort

Code Issues 24 Pull requests 0 Projects 0

github

SOCIAL CODING

GPU code for spike sorting

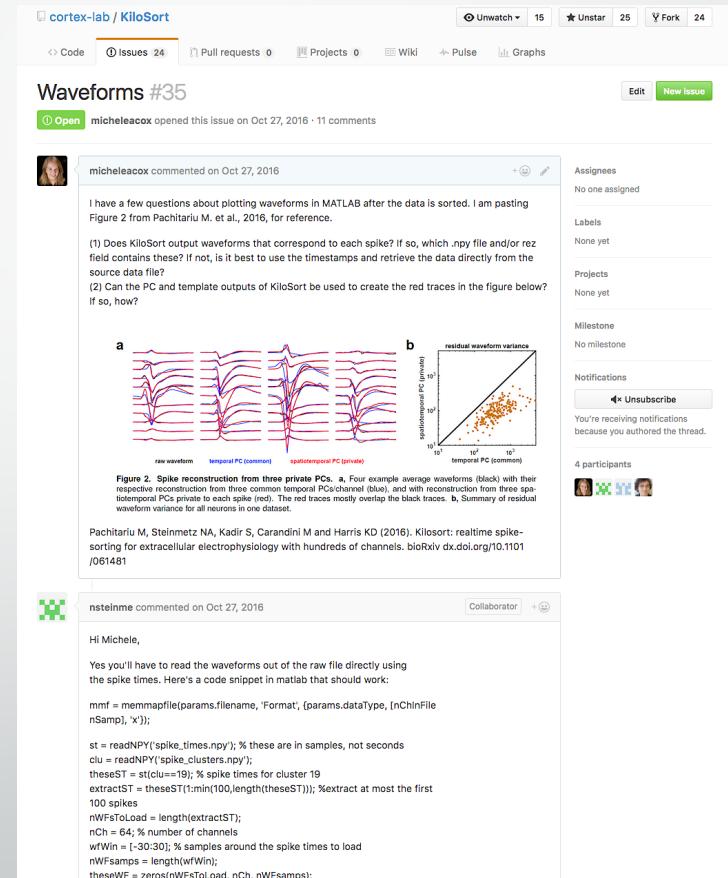
188 commits 9 branches 0 releases 5 contributors GPL-2.0

Branch: master New pull request

Create new file Upload files Find file Clone or download

marius10p committed on GitHub Update gather_mean_spikes.m Latest commit 99f264f 27 days ago

Got my questions answered!



cortex-lab / KiloSort

Unwatch 15 Unstar 25 Fork 24

Code Issues 24 Pull requests 0 Projects 0 Wiki Pulse Graphs

Waveforms #35

Open micheleaco opened this issue on Oct 27, 2016 · 11 comments

micheleaco commented on Oct 27, 2016

I have a few questions about plotting waveforms in MATLAB after the data is sorted. I am pasting Figure 2 from Pachitariu M. et al., 2016, for reference.

(1) Does KiloSort output waveforms that correspond to each spike? If so, which .npy file and/or rez field contains these? If not, is it best to use the timestamps and retrieve the data directly from the source data file?

(2) Can the PC and template outputs of KiloSort be used to create the red traces in the figure below? If so, how?

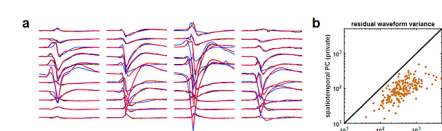


Figure 2. Spike reconstruction from three private PCs. a. Four example average waveforms (black) with their respective reconstruction from three common temporal PCs channel (blue), and with reconstruction from three spatiotemporal PCs private to each spike (red). The red traces mostly overlap the black traces. b. Summary of residual waveform variance for all neurons in one dataset.

Pachitariu M, Steinmetz NA, Kadir S, Carandini M and Harris KD (2016). Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. bioRxiv dx.doi.org/10.1101/061481

nsteinme commented on Oct 27, 2016 Collaborator

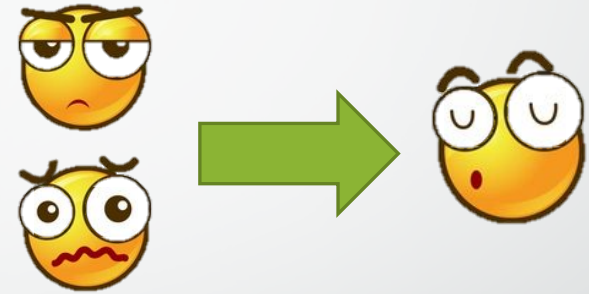
Hi Michele,

Yes you'll have to read the waveforms out of the raw file directly using the spike times. Here's a code snippet in matlab that should work:

```
mmf = memmapfile(params.filename, 'Format', [params.dataType, nChnFile nSamp, 'x']);  
  
st = readNPY('spike_times.npy'); % these are in samples, not seconds  
clu = readNPY('spike_clusters.npy');  
theseST = st(clu==19); % spike times for cluster 19  
extractST = theseST(1:min(100,length(theseST))); %extract at most the first 100 spikes  
nWfsToLoad = length(extractST);  
nCh = 64; % number of channels  
wFwin = [-30:30]; % samples around the spike times to load  
nWfsamps = length(wFwin);  
theseWF = zeros(nWfsToLoad, nCh, nWfsamps);
```

Objectives

- **Lower the barrier to entry** for people to start using Git and GitHub
 - How to get started
 - Technical demonstrations
 - Tackle resistance
- **Share** how I have used Git and GitHub in my research workflow
 - Encourage others to share their experiences using Git and share other approaches to version control and sharing that work for their labs



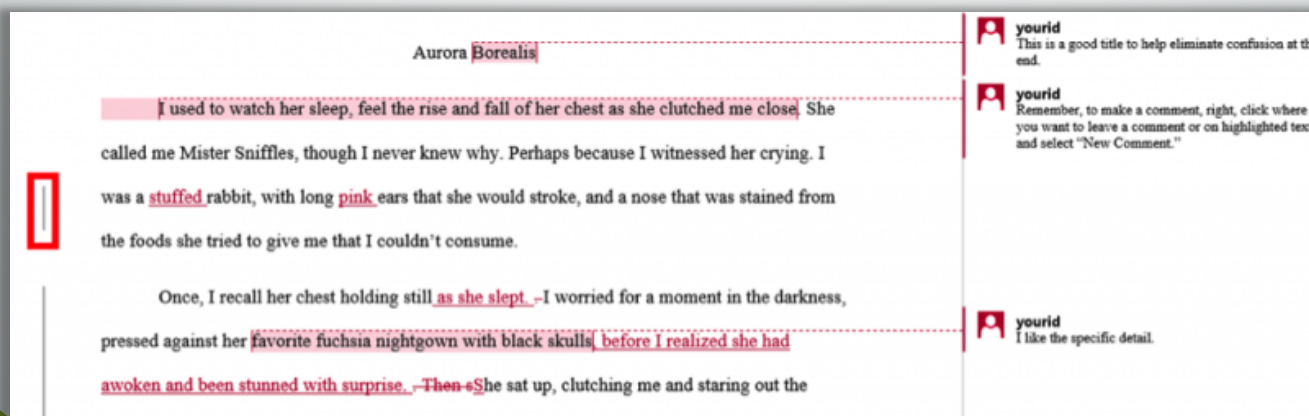
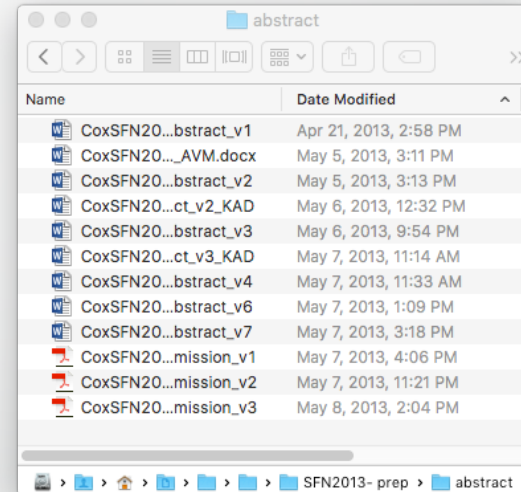
Pacing

- How does Git work?
 - 15 slides
 - 2 demos
- Collaborating with Git and GitHub
 - 15 slides
- My experiences using Git
 - 15 slides

What is Git?

version control software

- Simply, **version control is a way of logging changes to a file**
 - **What** was changed
 - **When** it was changed
 - **Who** changed it
- A lot of us already do this!



How does Git work? *on a conceptual Level*



Git, this is where I want you to keep track of files

```
>> Git INIT
```

- *shell/terminal*
- *GUI program*
- *Matalb 2014b+*

How does Git work?

on a conceptual Level



Git, this is a file
for which I want
you to track
changes into
the future

```
>> Git ADD "file"  
>> Git COMMIT
```

How does Git work?

on a conceptual Level



Git, I *made changes* to the file you were tracking.

```
>> Git ADD "file"  
>> Git COMMIT
```

How does Git work? *on a conceptual Level*

In Git, the process of logging changes
—including adding new files—
involves **2 steps*** (i.e., 2 commands):



```
>> Git ADD  
>> Git COMMIT
```

"commit"

**this is different from SVN*

[http://softwareengineering.stackexchange.com/questions/69178/
what-is-the-benefit-of-gits-two-stage-commit-process-staging](http://softwareengineering.stackexchange.com/questions/69178/what-is-the-benefit-of-gits-two-stage-commit-process-staging)

<https://www.atlassian.com/git/tutorials/saving-changes>

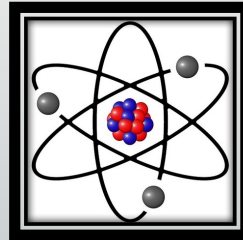
How does Git work? *on a conceptual Level*

Git **COMMIT**

- The main “workhorse” of the Git user



- The basic “change unit” in Git



How does Git work?

on a conceptual Level

Git **COMMIT**

- A commits most critical feature is that it **requires a message from the user**

```
$ git add file1.cpp  
$ git commit -m "Commit message"
```

- As a result, it creates an **annotated history** of changes over time



Demo 1

Basic Git Concepts

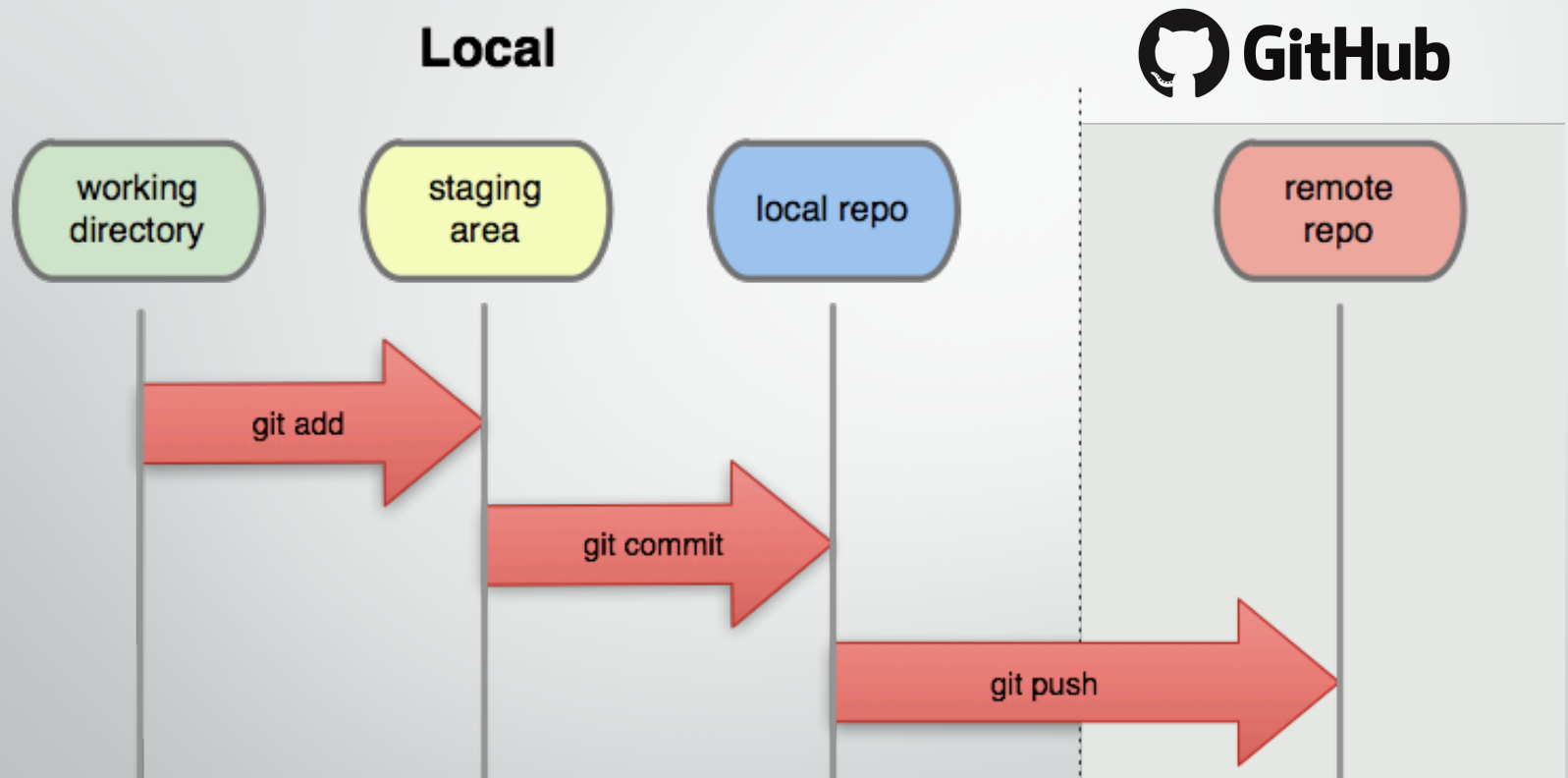
In Git, the process of logging changes involves 2 steps

- "ADD" the change
- "COMMIT" the change
 - ***with a message!***

This all happens locally




- Make changes in the *working directory*
- Add changes to the *staging area*
- Commit changes to the *repository*

Basic Git Concepts



In case of fire



-  1. git commit
-  2. git push
-  3. leave building

Demo 2

Demo 2 – Git Status

The screenshot shows the Git GUI interface for a repository named "ephys-analysis". The interface is divided into several sections:

- Top Bar:** Contains icons for Commit, Pull, Push, Fetch, Branch, Merge, and Stash. On the right, there are options for "Show in Finder", "Terminal", and "Settings".
- Left Sidebar:** Lists navigation options: WORKSPACE, File status, History, Search, BRANCHES (with "master" selected), TAGS, REMOTES (with "origin" selected), STASHES, SUBMODULES, and SUBTREES.
- Commit History Table:** A table showing the commit history. The current commit is highlighted in blue.

Graph	Description	Commit	Author	Date
o	Uncommitted changes	*	*	Today, 3:45 PM
o	vargout and ss.clusterMap development	1393b32	Michele Cox <michele.a.cox@gmail.com>	Jan 14, 2017, 3:29:49 PM CST
o	added some plot examples to comments	174e2ae	Michele Cox <michele.a.cox@gmail.com>	Jan 14, 2017, 3:12:00 PM CST
o	small change to previous commit	6384271	Michele Cox <michele.a.cox@gmail.com>	Jan 14, 2017, 3:12:00 PM CST
o	added vargout	730bc48	Michele Cox <michele.a.cox@gmail.com>	Jan 14, 2017, 2:50:00 PM CST
o	big change! added mapping from cluster ID to channel and a automatic rejection criteria	a2f0366	Michele Cox <michele.a.cox@gmail.com>	Jan 14, 2017, 2:50:00 PM CST
o	continued to work on option for output/summary plots	92307cf	Michele Cox <michele.a.cox@gmail.com>	Jan 11, 2017, 3:03:00 PM CST
o	added varargout assignment for summary info	60fda95	Michele Cox <michele.a.cox@gmail.com>	Jan 11, 2017, 12:10:00 PM CST
o	added line items for new analyses	5b78e10	Michele Cox <michele.a.cox@gmail.com>	Jan 11, 2017, 10:40:00 PM CST
o	small changes	f6bdc3b	Michele Cox <michele.a.cox@gmail.com>	Jan 6, 2017, 7:23:00 PM CST
o	added photodiode triggering and psth/latency analysis	582923d	Michele Cox <michele.a.cox@gmail.com>	Jan 6, 2017, 7:23:00 PM CST
o	new code for triggering to photodiode for STATIC gratings only (Kacie has code for drifting).	3a0830e	Michele Cox <michele.a.cox@gmail.com>	Jan 6, 2017, 7:22:00 PM CST
- Diff View:** Shows the changes in the file "runAlignmentPDFs/getWavebyDepth.m". A hunk of lines 118-133 is selected, showing a case statement for 'KiloSorted'. The diff highlights changes with red (removal) and green (addition) backgrounds.

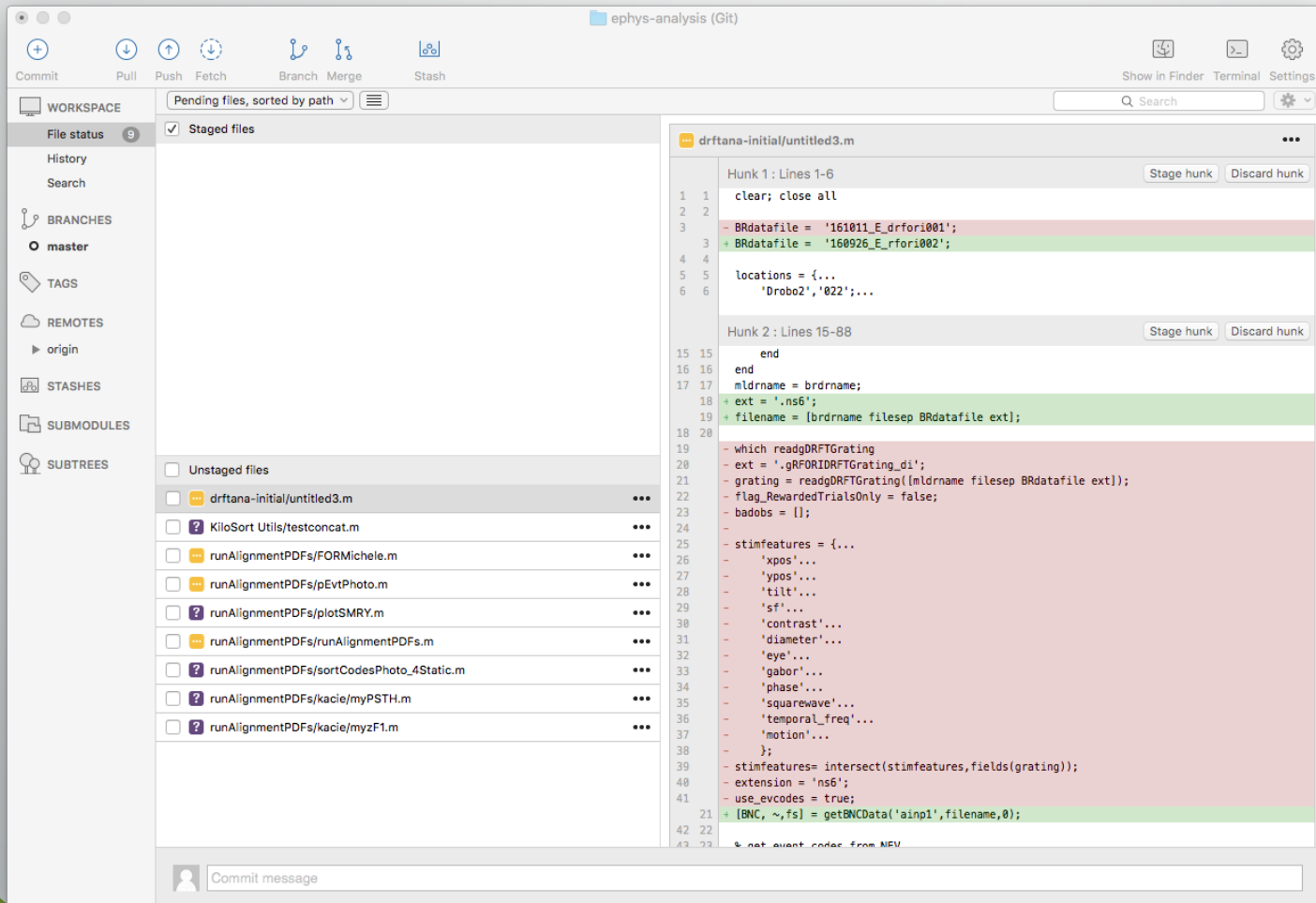
```
Hunk 1 : Lines 118-133
Reverse hunk

118 118
119 119
120 120
121 - case 'KiloSorted'
122 -     eid = find(strcmp(ss.chanIDs,elabel));
123 -     if isempty(eid)
124 -         continue
125 -     end
126 -     I1 = ss.peakSpikeCh == eid;
127 -     I2 = ss.peakTempCh == eid;
128 -     I3 = ss.peakWaveCh == eid;
129 -     possible_clusters = ss.spikeClusters(I1 | I2 | I3);
130 -     if isempty(possible_clusters) ...
131 -         || length(unique(possible_clusters)) > 3 ...
132 -         || numel(possible_clusters)<10;
133 -         continue
134 -     end
135 -     clustID = mode(possible_clusters);
136 -     I = ss.spikeClusters == clustID;
137 -     clustID = ss.clusterMap(cidx,1);
138 -     I = ss.spikeClusters == clustID;
139 -     WAVE = squeeze(ss.spikeWaves(eid,:,I));
140 - end

121 +     eid = find(strcmp(ss.chanIDs,elabel));
122 +     if isempty(eid)
123 +         continue
124 +     end
125 +     cidx = find(ss.clusterMap(:,2) == eid,1,'first');
126 +     if isempty(cidx) || ss.clusterMap(cidx,end) == 0
127 +         continue
128 +     end
129 +     clustID = ss.clusterMap(cidx,1);
130 +     I = ss.spikeClusters == clustID;
```
- Commit Details:** Shows details for the current commit: "vargout and ss.clusterMap development".

Commit: 1393b32fe311ecc5c2cbbf8f61dacbd4...
Parents: 174e2aecba
Author: Michele Cox <michele.a.cox@gmail.com>
Date: January 14, 2017 at 3:29:49 PM CST
Labels: HEAD -> master origin/master origin/H...

Demo 2 – Staging Area



The screenshot displays the Git GUI interface for a repository named "ephys-analysis (Git)". The top toolbar includes buttons for Commit, Pull, Push, Fetch, Branch, Merge, and Stash. The left sidebar shows the workspace structure with sections for File status, History, Search, BRANCHES (master), TAGS, REMOTES (origin), STASHES, SUBMODULES, and SUBTREES. The main area is divided into two panes. The left pane, titled "Pending files, sorted by path", shows the "Staged files" section with a list of files, including "drftana-initial/untitled3.m". The right pane shows a diff view for the file "drftana-initial/untitled3.m", highlighting changes in two hunks. Hunk 1 (Lines 1-6) shows a change in the "BRdatafile" variable. Hunk 2 (Lines 15-88) shows a change in the "filename" variable and the addition of a new line for "[BNC, ~,fs]".

```
Hunk 1: Lines 1-6
1 1 clear; close all
2 2
3 - BRdatafile = '161011_E_drfori001';
3 + BRdatafile = '160926_E_rfori002';
4 4
5 5 locations = {...
6 6     'Drobo2', '022';...

Hunk 2: Lines 15-88
15 15 end
16 16 end
17 17 mldrname = brdrname;
18 + ext = '.ns6';
19 + filename = [brdrname filesep BRdatafile ext];
18 20
19 - which readDRFTGrating
20 - ext = 'gRFORIDRFTGrating_di';
21 - grating = readDRFTGrating([mldrname filesep BRdatafile ext]);
22 - flag_RewardedTrialsOnly = false;
23 - badobs = [];
24 -
25 - stimfeatures = {...
26 -     'xpos'...
27 -     'ypos'...
28 -     'tilt'...
29 -     'sf'...
30 -     'contrast'...
31 -     'diameter'...
32 -     'eye'...
33 -     'gabor'...
34 -     'phase'...
35 -     'squarewave'...
36 -     'temporal_freq'...
37 -     'motion'...
38 - };
39 - stimfeatures= intersect(stimfeatures,fields(grating));
40 - extension = 'ns6';
41 - use_evcodes = true;
42 + [BNC, ~,fs] = getBNCData('ainp1',filename,0);
42 22
43 23 % get event codes from NEV
```

Demo 2 – Git Add, Commit, Push

The screenshot displays the Git GUI interface for a repository named 'ephys-analysis (Git)'. The workspace shows a file 'runAlignmentPDFs/plotSMRY.m' staged for commit. The diff view shows changes to 'drftana-initial/untitled3.m', including adding 'BRdatafile' and 'filename' fields. The commit message is 'initial version of summary plot'.

Workspace: Pending files, sorted by path

File status: Staged files

- runAlignmentPDFs/plotSMRY.m

Unstaged files:

- drftana-initial/untitled3.m
- KiloSort Utils/testconcat.m
- runAlignmentPDFs/FORMichele.m
- runAlignmentPDFs/pEvtPhoto.m
- runAlignmentPDFs/runAlignmentPDFs.m
- runAlignmentPDFs/sortCodesPhoto_4Static.m
- runAlignmentPDFs/kacie/myPSTH.m
- runAlignmentPDFs/kacie/myzF1.m

Diff view (drftana-initial/untitled3.m):

```
Hunk 1: Lines 1-6
1 1 clear; close all
2 2
3 - BRdatafile = '161011_E_drfori001';
3 + BRdatafile = '160926_E_rfori002';
4 4
5 5 locations = {...
6 6 'Drobo2', '022';...

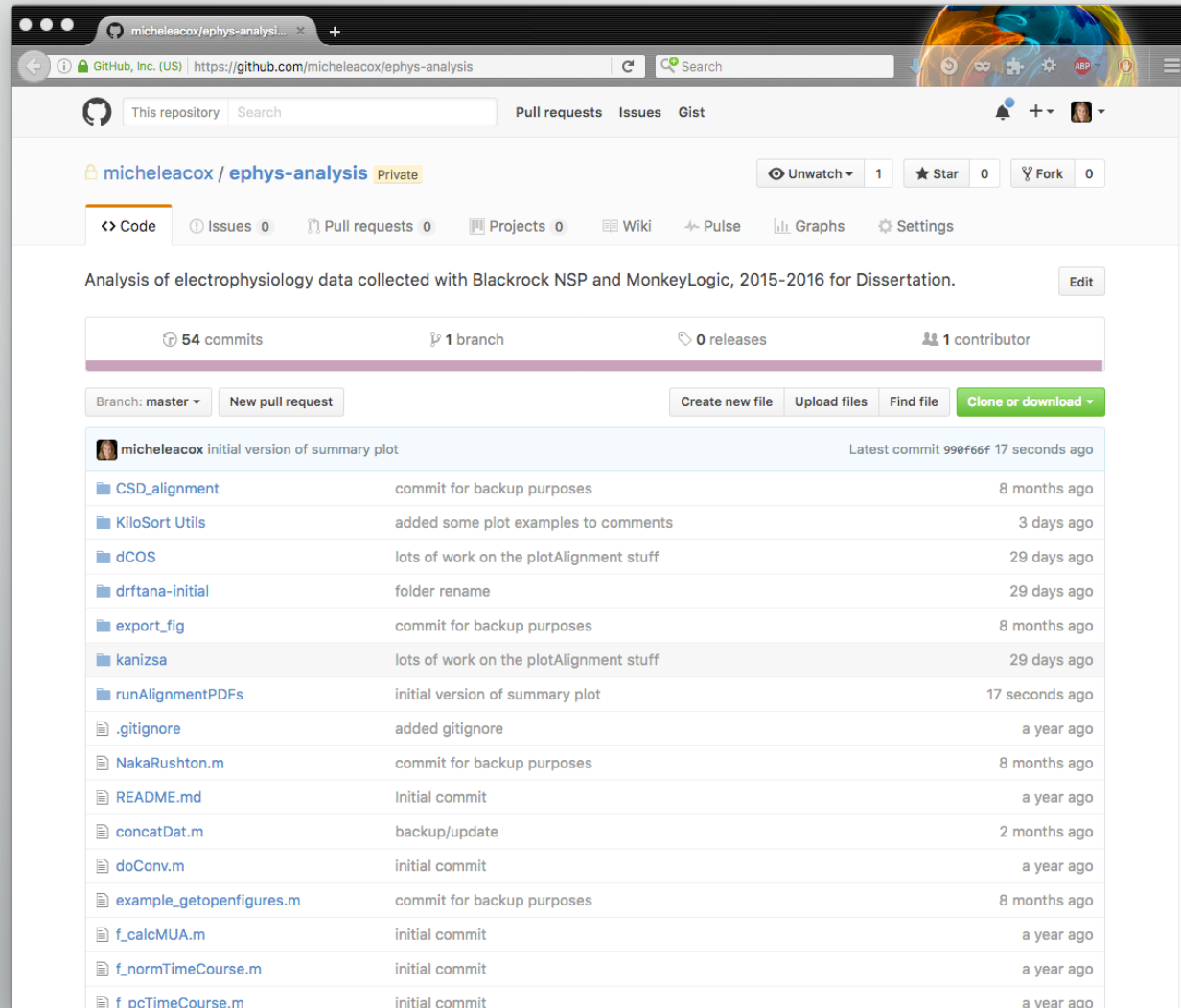
Hunk 2: Lines 15-88
15 15 end
16 16 end
17 17 mldrname = brdrname;
18 + ext = '.ns6';
19 + filename = [brdrname filesep BRdatafile ext];
20 20
21 - which readgDRFTGrating
22 - ext = 'gRFORIDRFTGrating_di';
23 - grating = readgDRFTGrating([mldrname filesep BRdatafile ext]);
24 - flag_RewardedTrialsOnly = false;
25 - badobs = [];
26 26
27 - stimfeatures = {...
28 - 'xpos'...
29 - 'ypos'...
30 - 'tilt'...
31 - 'sf'...
32 - 'contrast'...
33 - 'diameter'...
34 - 'eye'...
35 - 'gabor'...
36 - 'phase'...
37 - 'squarewave'...
38 - 'temporal_freq'...
39 - 'motion'...
```

Commit message: Michele Cox <michele.a.cox@gmail.com> initial version of summary plot

Options: Push changes immediately to origin/master

Buttons: Cancel, Commit

Demo 2 – Remote GitHub Repo



The screenshot shows a GitHub repository page for 'micheleacox / ephys-analysis'. The repository is private and has 1 star, 0 forks, and 0 pull requests. The description is 'Analysis of electrophysiology data collected with Blackrock NSP and MonkeyLogic, 2015-2016 for Dissertation.' The repository has 54 commits, 1 branch, 0 releases, and 1 contributor. The current branch is 'master'. The file list includes folders like 'CSD_alignment', 'KiloSort Utils', 'dCOS', 'drftana-initial', 'export_fig', 'kanizsa', and 'runAlignmentPDFs', as well as files like '.gitignore', 'NakaRushton.m', 'README.md', 'concatDat.m', 'doConv.m', 'example_getopenfigures.m', 'f_calcMUA.m', 'f_normTimeCourse.m', and 'f_pcTimeCourse.m'.

Repository: micheleacox / ephys-analysis Private

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

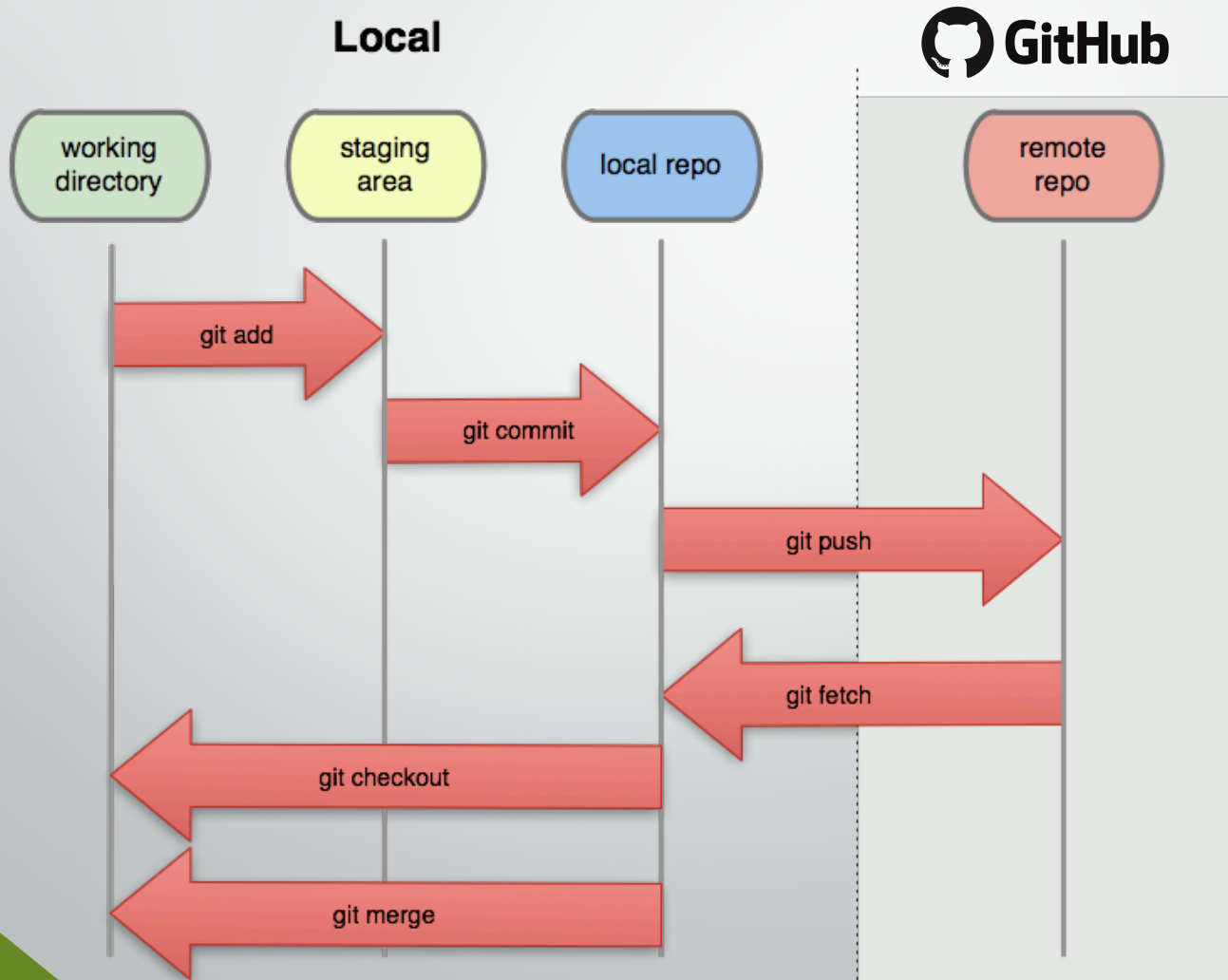
Analysis of electrophysiology data collected with Blackrock NSP and MonkeyLogic, 2015-2016 for Dissertation. Edit

54 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

File/Folder	Commit Message	Time Ago
micheleacox	initial version of summary plot	Latest commit 990f66f 17 seconds ago
CSD_alignment	commit for backup purposes	8 months ago
KiloSort Utils	added some plot examples to comments	3 days ago
dCOS	lots of work on the plotAlignment stuff	29 days ago
drftana-initial	folder rename	29 days ago
export_fig	commit for backup purposes	8 months ago
kanizsa	lots of work on the plotAlignment stuff	29 days ago
runAlignmentPDFs	initial version of summary plot	17 seconds ago
.gitignore	added gitignore	a year ago
NakaRushton.m	commit for backup purposes	8 months ago
README.md	Initial commit	a year ago
concatDat.m	backup/update	2 months ago
doConv.m	initial commit	a year ago
example_getopenfigures.m	commit for backup purposes	8 months ago
f_calcMUA.m	initial commit	a year ago
f_normTimeCourse.m	initial commit	a year ago
f_pcTimeCourse.m	initial commit	a year ago

Basic Git Concepts

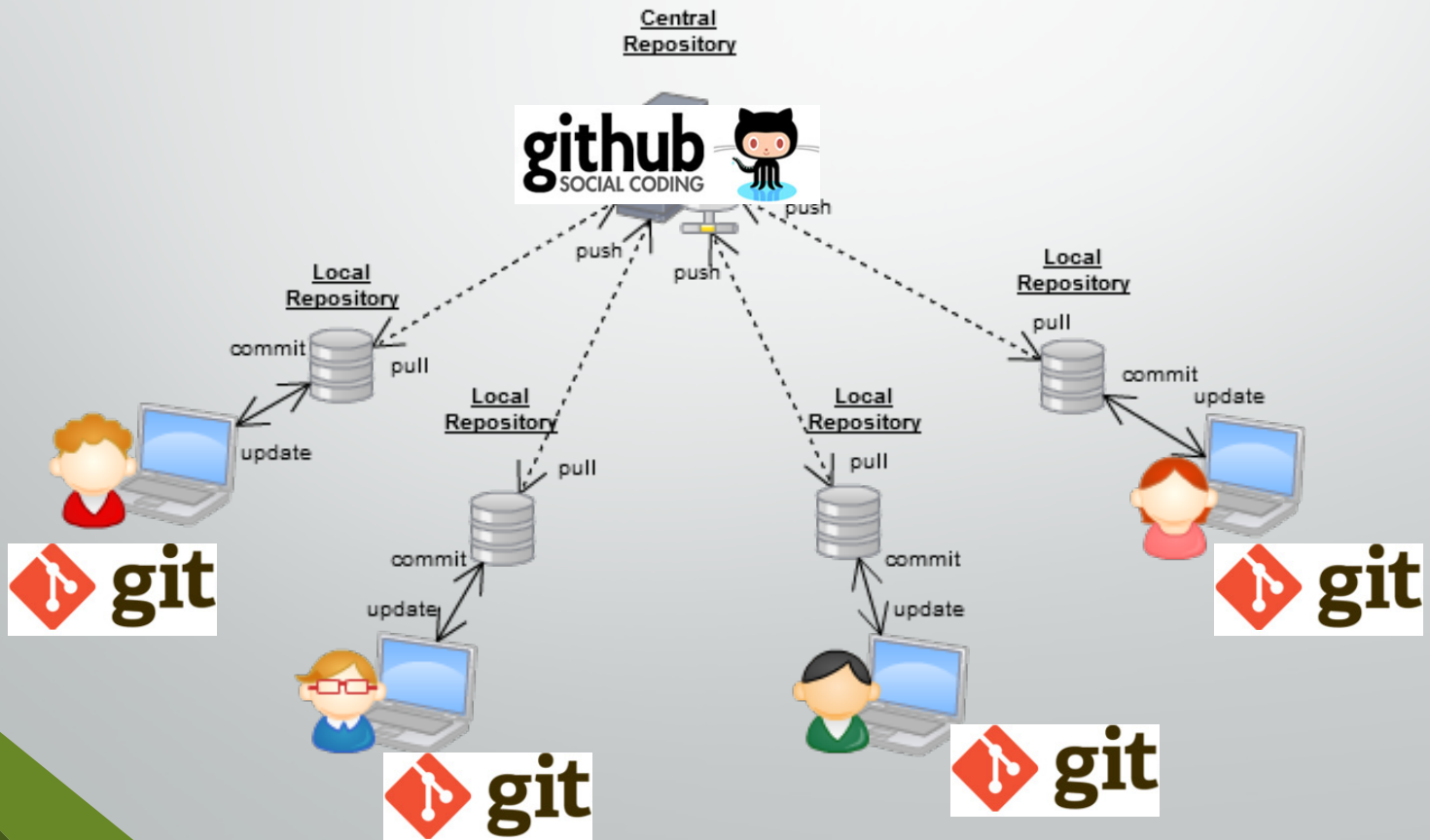


Basic Git Concepts

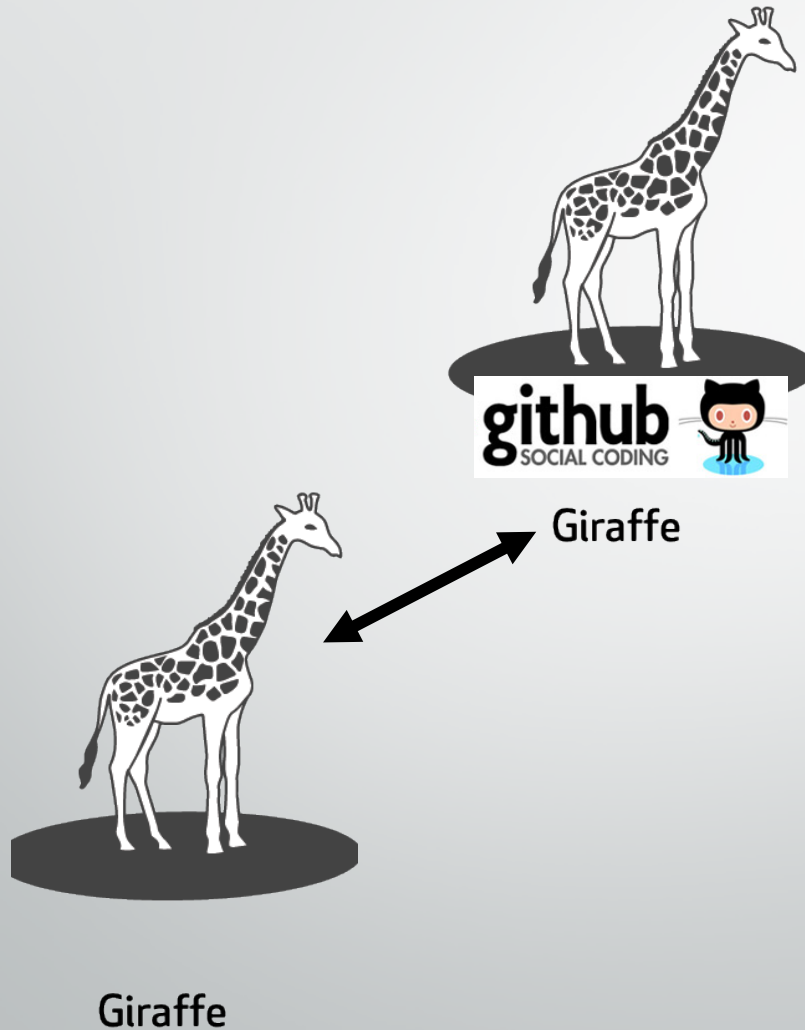
- Key instruments:
 - ADD – tells Git that **you have made a change**
 - COMMIT – tells Git to **log changes** with a message
 - PUSH – tells Git to **transfer those changes** to remote
- Commits contain:
 - **A record of changes** (line by line, new items, etc.)
 - **Your notes** (i.e., message) about the change
 - **Date/time/person** making the commit.

Collaborating with Git

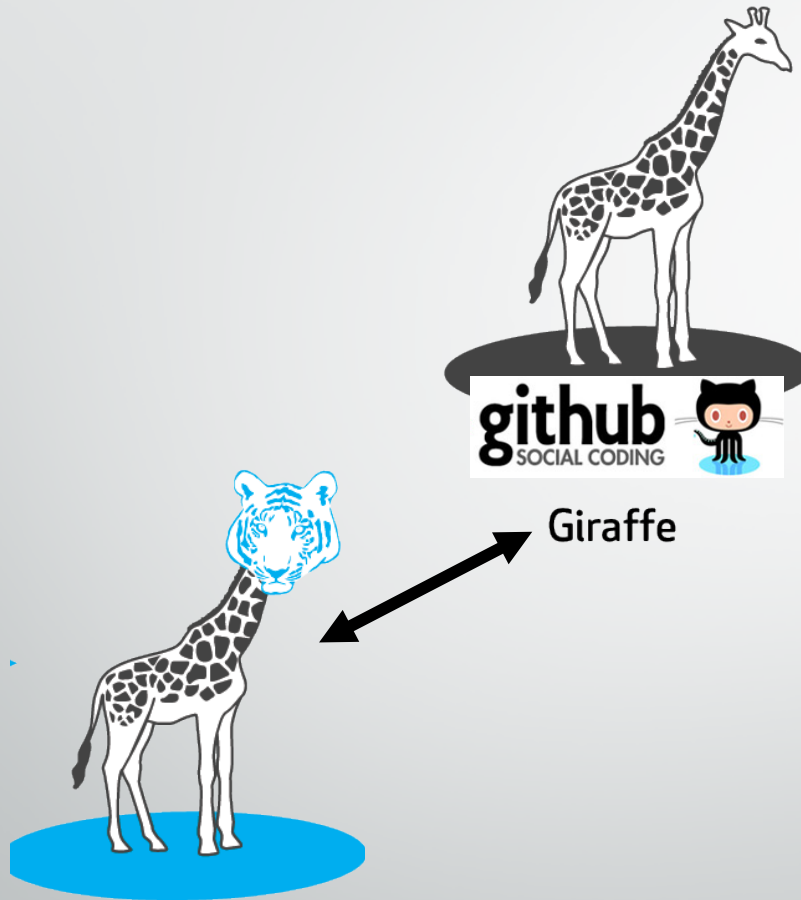
- Commits are the basic “change unit” in Git
- Commits are also the basic unit of collaboration



Collaborating with Git

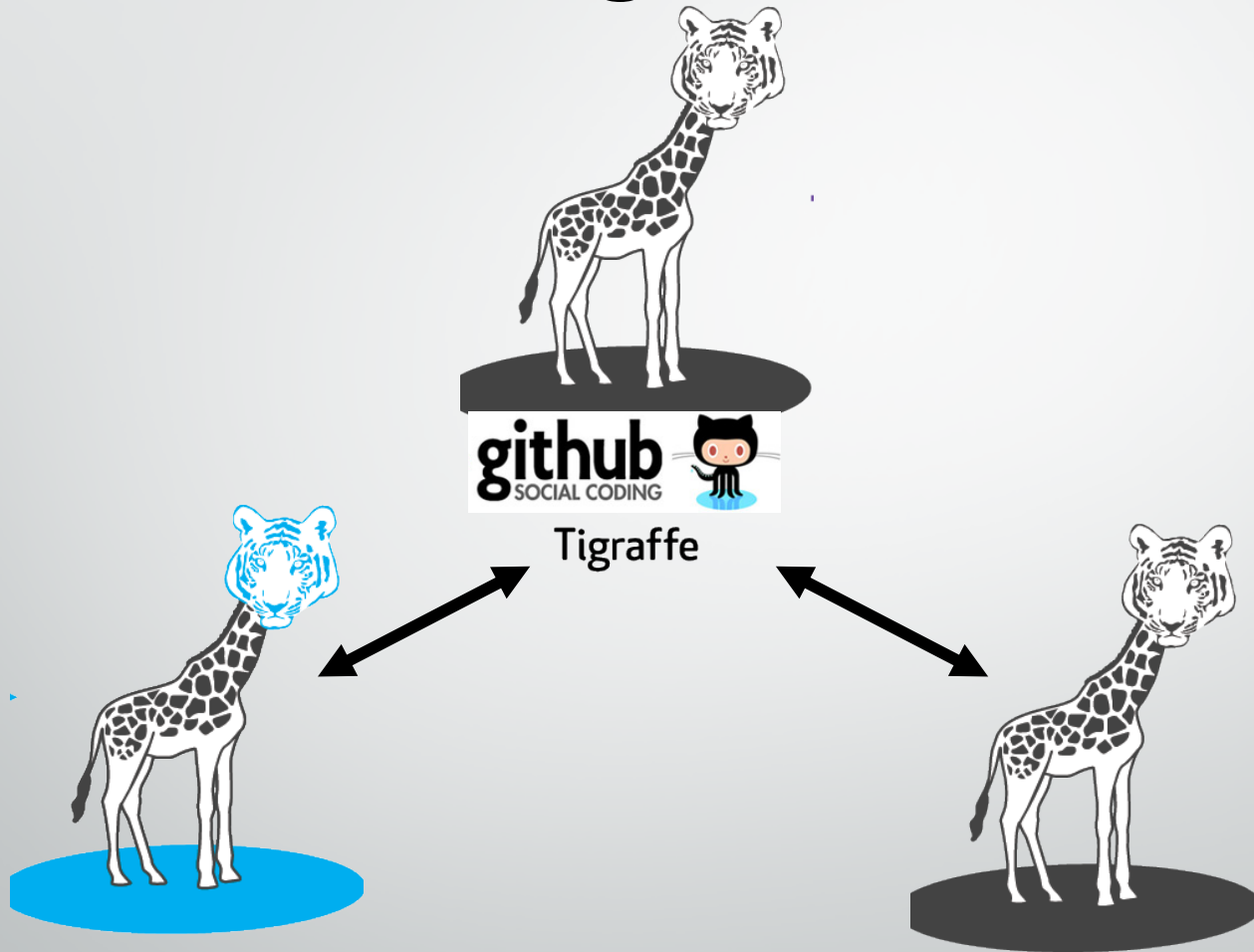


Collaborating with Git



Jean adds
tiger head

Collaborating with Git

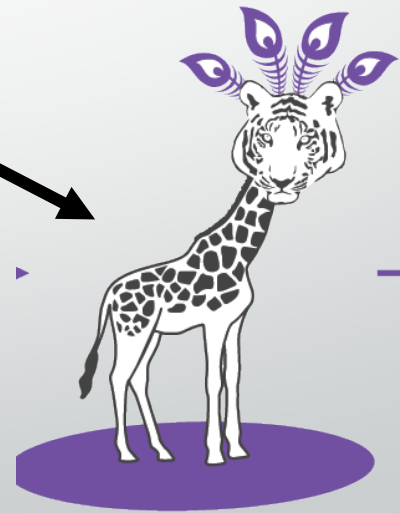


Jean adds
tiger head

Collaborating with Git

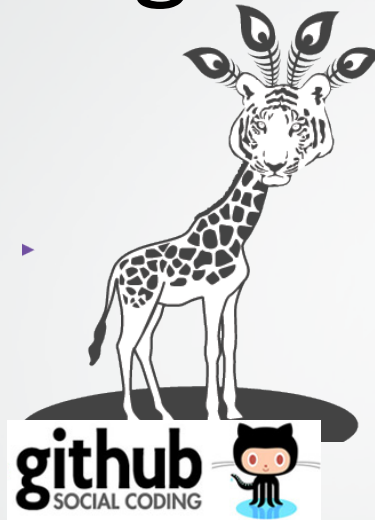


Tigraffe

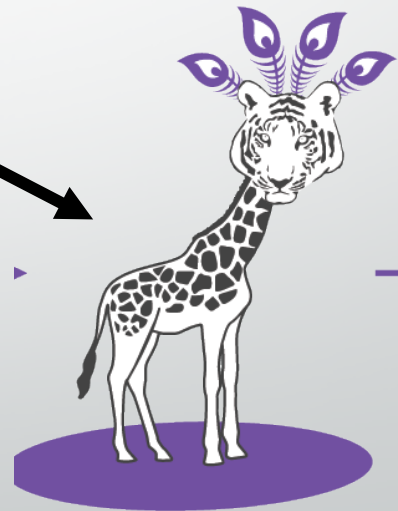


Sam adds
peacock feathers

Collaborating with Git

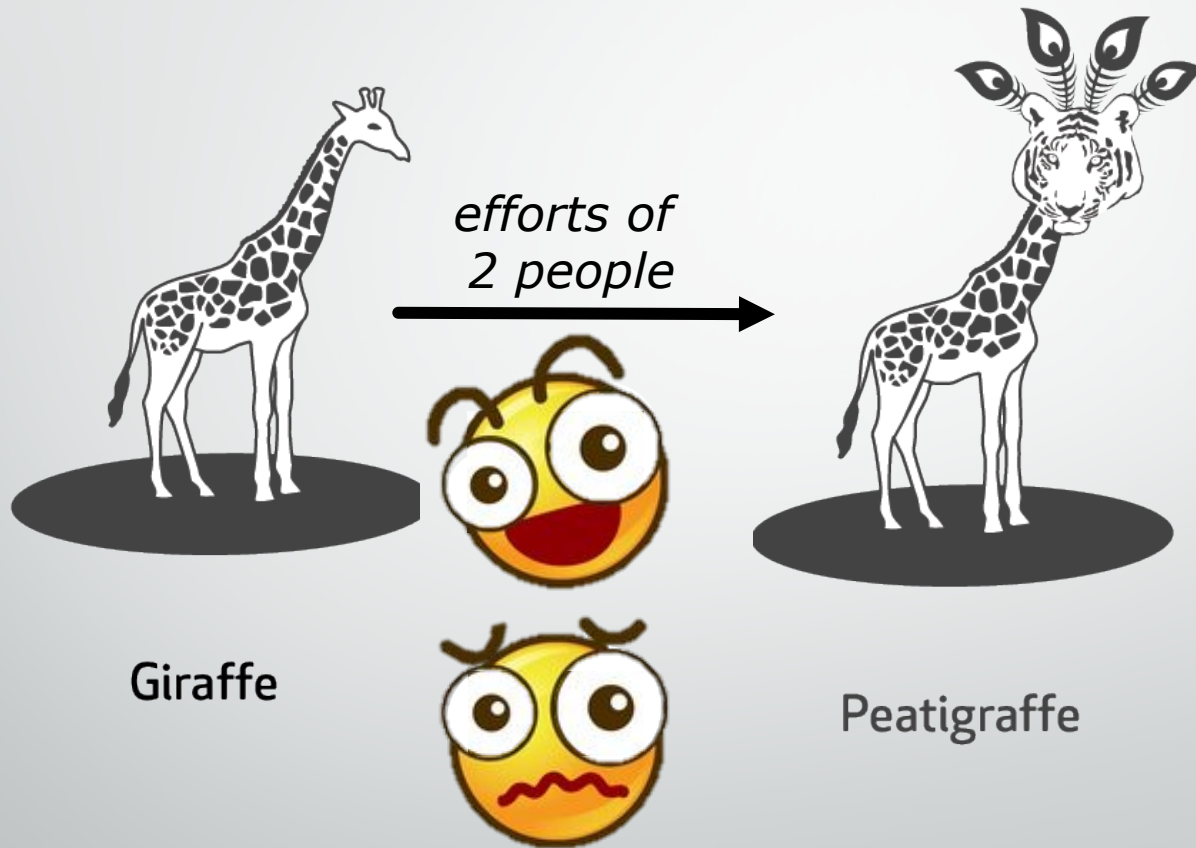


Peatigraffe



Sam adds
peacock feathers

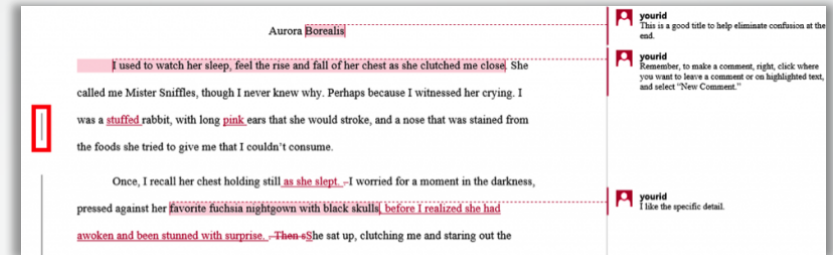
Collaborating with Git



Collaborating with Git



- You can imagine how this system might go awry
 - Maybe you don't want a Petigraffe
- Loss of control
 - What if somebody changes something badly



- Remember, **Git tracks all changes** on a line-by-line bases
 - You can always reverse or revert a change
- Git contains a **variety of merge tools and safety checks**
 - MERGE CONFLICT -- Will tell you if there is a conflict between your another person's commits

Collaborating with Git

Merge Conflicts

🏠 Don't be afraid to commit

latest

Search docs

Docs » Git and GitHub » Resolving conflicts

[Edit on GitHub](#)

Resolving conflicts

In this section you will:

- encounter a merge conflict on GitHub
- encounter a merge conflict on the commandline
- resolve the conflict in a new temporary Git branch

Encountering a merge conflict on GitHub

Sometimes you'll discover that your GitHub fork and the upstream repository have changes that GitHub can't merge.

There's an *unmergeable-branch* at <https://github.com/evildmp/afraid-to-commit>. It's unmergeable because it deliberately contains changes that conflict with other changes made in *master*.

Using the GitHub interface, try creating a pull request from *unmergeable-branch* to your *master* on GitHub. If you do, GitHub will tell you:

We can't automatically merge this pull request.

Use the command line to resolve conflicts before continuing.

GitHub will in fact tell you the steps you need to take to solve this, but to understand what's actually happening, and to do it yourself when you need to, we need to cover some important concepts.

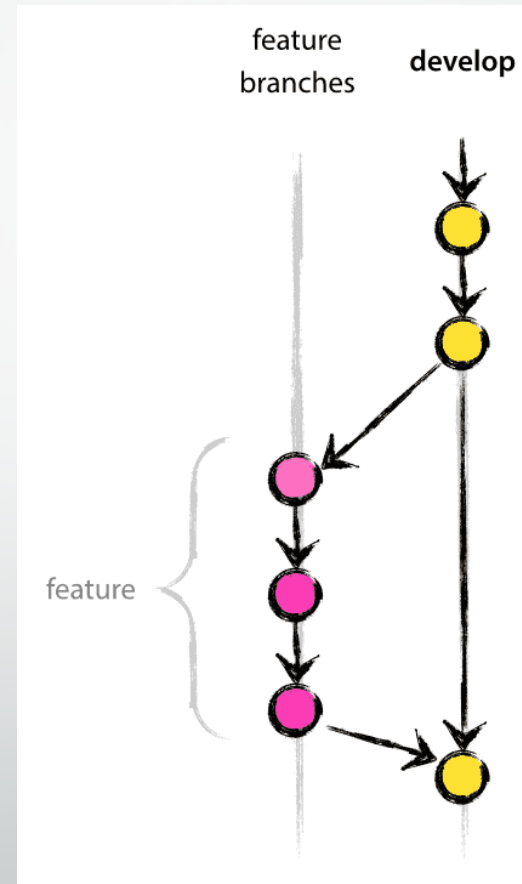
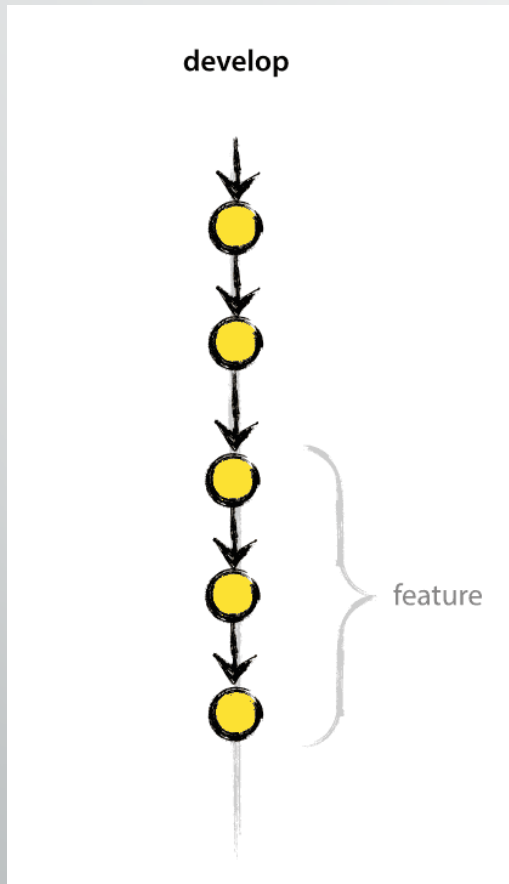
Merging changes from a remote branch

```
# make sure you have the latest data from upstream
$ git fetch upstream
# create and switch to a new branch based on master to explore the conflict
$ git checkout -b explore-conflict upstream/master
```

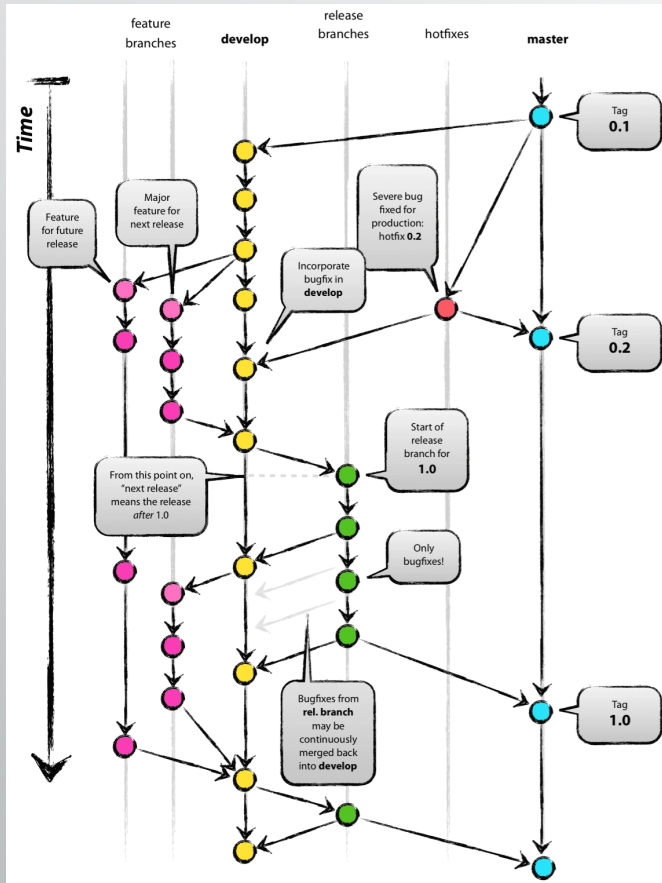
Collaborating with Git

- Git's key collaboration instruments are:
 - **Branching / Forking**
 - **Merging**

Collaborating with Git



Collaborating with Git



- Can have **many branches** for different purposes
- Can **switch** between them during development
- Can **push** to a remote repository without effecting other branches
- Eventually, can **merge** back into a main branch (often called the master branch).
- **Forks** allow all the same, but with repositories you don't control



This is where I ended

Getting Started

- Download and Install Git: <https://git-scm.com/>
 - Setup your first repository (git init)
 - Add and commit your first file
- Sign up for GitHub: <https://github.com/>
 - Setup a GitHub repository as the remote for your local repo
 - Push to the remote
- Optional
 - Request a free educational upgrade for GitHub as explained here: <http://www.inferencelab.com/free-github-private-repos-for-academics/>
 - Download SourceTree: <https://www.sourcetreeapp.com/>
 - Download DiffMerge: <https://sourcegear.com/diffmerge/>

“Try Git” simulator

*Resources for learning
Git and GitHub*

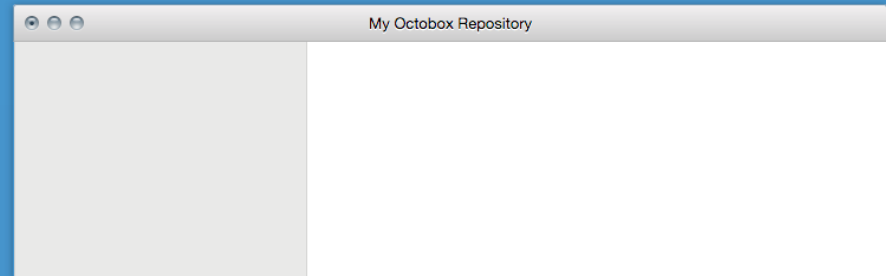
1.1 · Got 15 minutes and want to learn Git?

Git allows groups of people to work on the same documents (often code) at the same time, and without stepping on each other's toes. It's a distributed version control system.

Our terminal prompt below is currently in a directory we decided to name "octobox". To initialize a Git repository here, type the following command:

→ `git init`

tryGit



<https://try.github.io>

Getting Started

[About GitHub](#) / [Git and GitHub learning resources](#)

Git and GitHub learning resources

There are a lot of helpful Git and GitHub resources on the web. This is a short list of our favorites!

Using Git

Familiarize yourself with Git by visiting the [official Git project site](#) and reading the [ProGit ebook](#). You can review the [Git command list](#) or [Git command lookup reference](#) while using the [Try Git](#) simulator.

Using GitHub

Become better acquainted with GitHub through our [bootcamp](#) articles. See our [GitHub flow](#) for a process introduction. Refer to our [overview guides](#) to walk through basic concepts.

Branches, forks, and pull requests

Learn about [Git Branching](#) using an interactive tool. Read about [forks](#) and [pull requests](#) as well as [how we use pull requests](#) at GitHub.

Access quick references about the [command line](#) as well as GitHub [checklists](#), [cheat sheets](#), and [more](#).

Tune in

Our GitHub [YouTube Training and Guides channel](#) offers tutorials about [pull requests](#), [forking](#), [rebase](#), and [reset](#) functions. Each topic is covered in 5 minutes or less.


Windows users can view a special 10-minute [GitHub for Windows](#) tutorial presented by GitHub and Microsoft.

<https://help.github.com/articles/git-and-github-learning-resources/>

“Hello World” Demo on GitHub

*Resources for learning
Git and GitHub*

GitHub Guides Video Guides



The header of the GitHub guide features a dark red background with a subtle grid pattern. On the left is a circular icon containing a globe. To the right of the icon, the title "Hello World" is displayed in a large, white, sans-serif font. Below the title, a smaller white font indicates a "10 minute read" duration, preceded by a clock icon.

Hello World

🕒 10 minute read

The **Hello World** project is a time-honored tradition in computer programming. It is a simple exercise that gets you started when learning something new. Let's get started with GitHub!

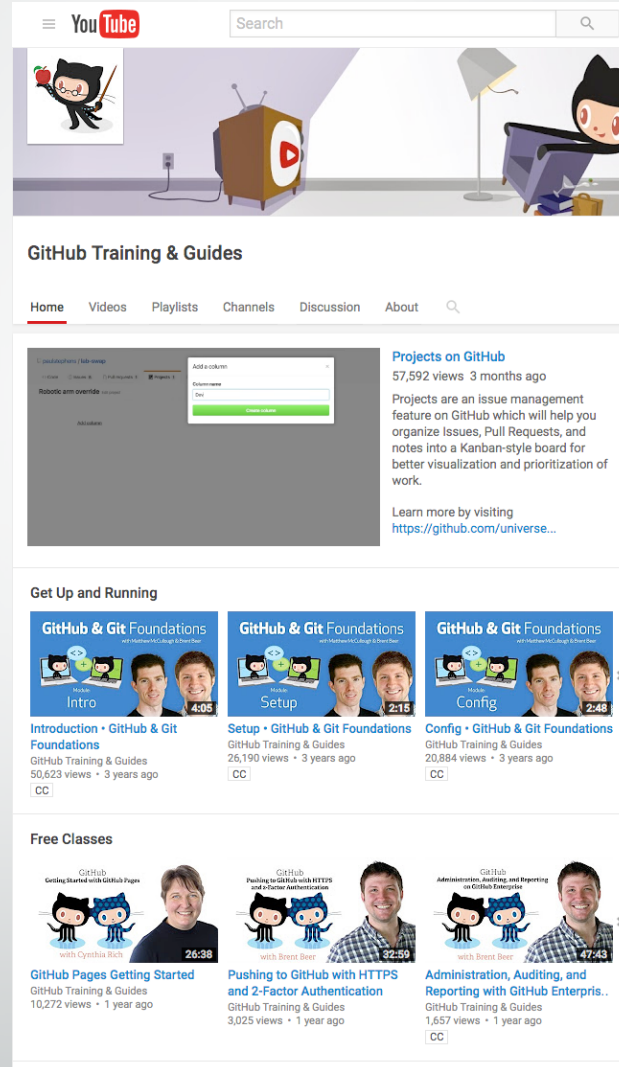
You'll learn how to:

- Create and use a repository
- Start and manage a new branch
- Make changes to a file and push them to GitHub as commits
- Open and merge a pull request

<https://guides.github.com/activities/hello-world/>

“GitHub” Channel on YouTube

*Resources for learning
Git and GitHub*



The screenshot shows the GitHub Training & Guides YouTube channel page. At the top, there is a search bar and a navigation menu with options: Home, Videos, Playlists, Channels, Discussion, and About. Below the navigation, there is a featured video titled "Projects on GitHub" with 57,592 views, posted 3 months ago. The video description explains that Projects are an issue management feature on GitHub that help organize issues, pull requests, and notes into a Kanban-style board. A link is provided to learn more: <https://github.com/universe...>

Below the featured video, there are three video thumbnails under the heading "Get Up and Running":

- Intro**: 4:05 duration, 50,623 views, 3 years ago.
- Setup**: 2:15 duration, 26,190 views, 3 years ago.
- Config**: 2:48 duration, 20,884 views, 3 years ago.

Each of these videos is part of the "Introduction • GitHub & Git Foundations" playlist.

Below the "Get Up and Running" section, there are three video thumbnails under the heading "Free Classes":

- GitHub Pages Getting Started**: 26:38 duration, 10,272 views, 1 year ago.
- Pushing to GitHub with HTTPS and 2-Factor Authentication**: 32:59 duration, 3,025 views, 1 year ago.
- Administration, Auditing, and Reporting with GitHub Enterprise..**: 47:43 duration, 1,657 views, 1 year ago.

Each of these videos is part of the "GitHub Training & Guides" playlist.

<https://www.youtube.com/githubguides>

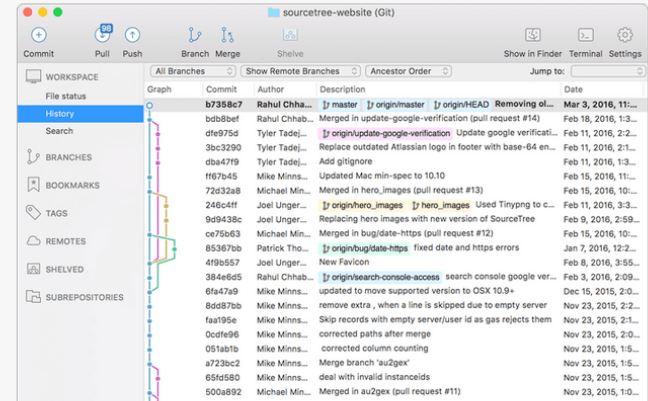
Source Tree

- Git and GitHub can be used entirely from the command line.
- But, there are many GUI implementations of Git. The one I use is called SourceTree
- Personally, I'm not a fan of the GitHub desktop application

Harness the power of Git and Hg in a beautifully simple application

Download for Mac OS X

Also available for Windows



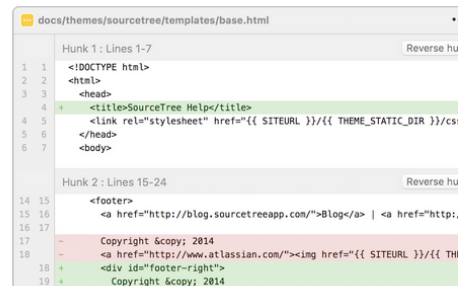
Join the SourceTree Beta Program

Want access to new features and improvements before they are in production? Sign up for the Beta program to try new features, provide feedback and engage with the SourceTree Team.

Sign up now

A free visual Git and Hg client for Mac and Windows

SourceTree simplifies how you interact with your Git and Mercurial repositories so you can focus on coding. Visualize and manage your repositories through SourceTree's simple interface.



Simple for beginners

Say goodbye to the command line - simplify distributed version control for your team and quickly bring everyone up to speed.

Powerful for experts

Perfect for making advanced users even more productive. Review changesets, stash, cherry-pick between branches and more.