

Using Git Flow to Manage Code Projects with GitKraken

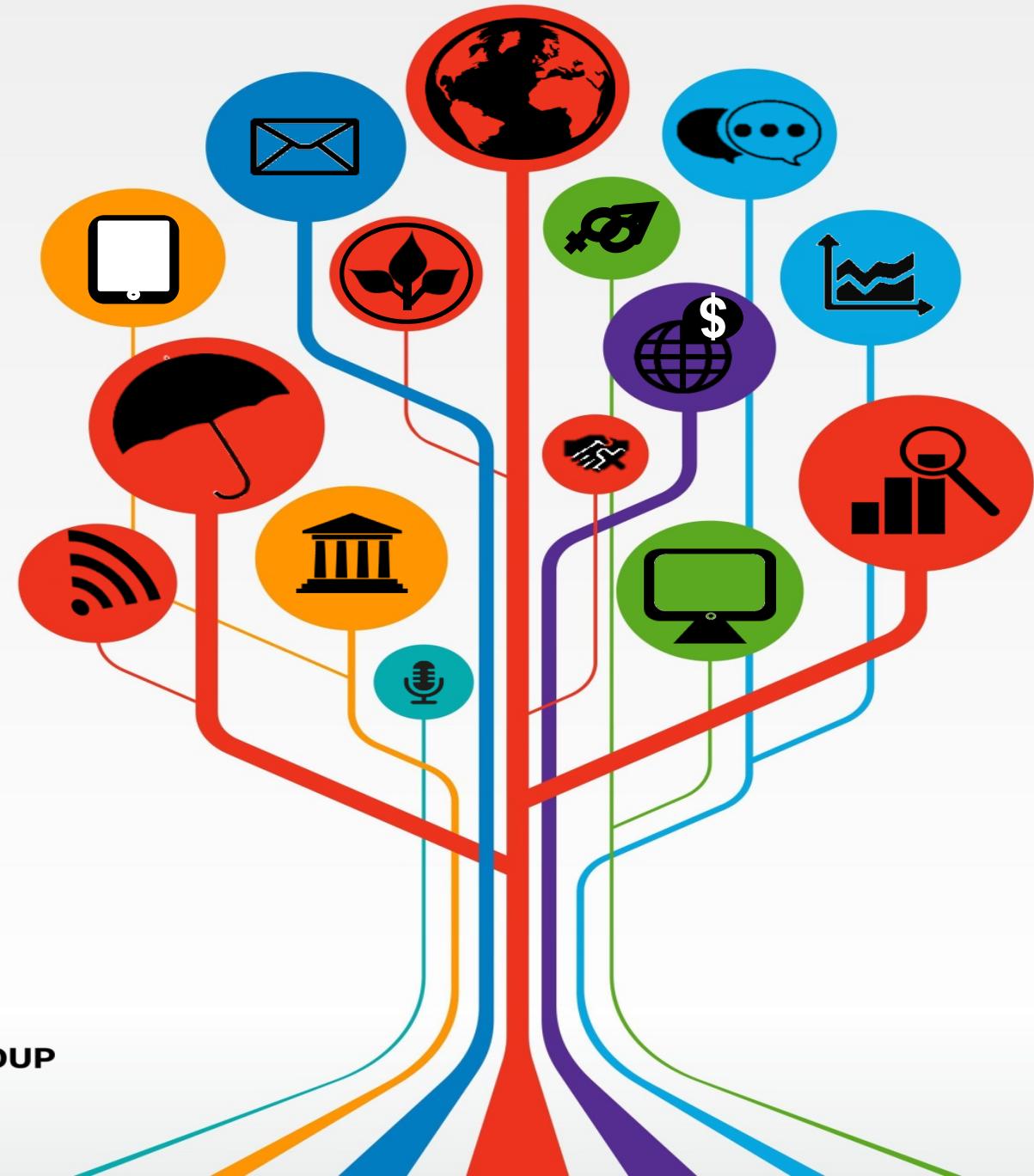
Research Assistant Onboarding

Prepared by DIME Analytics

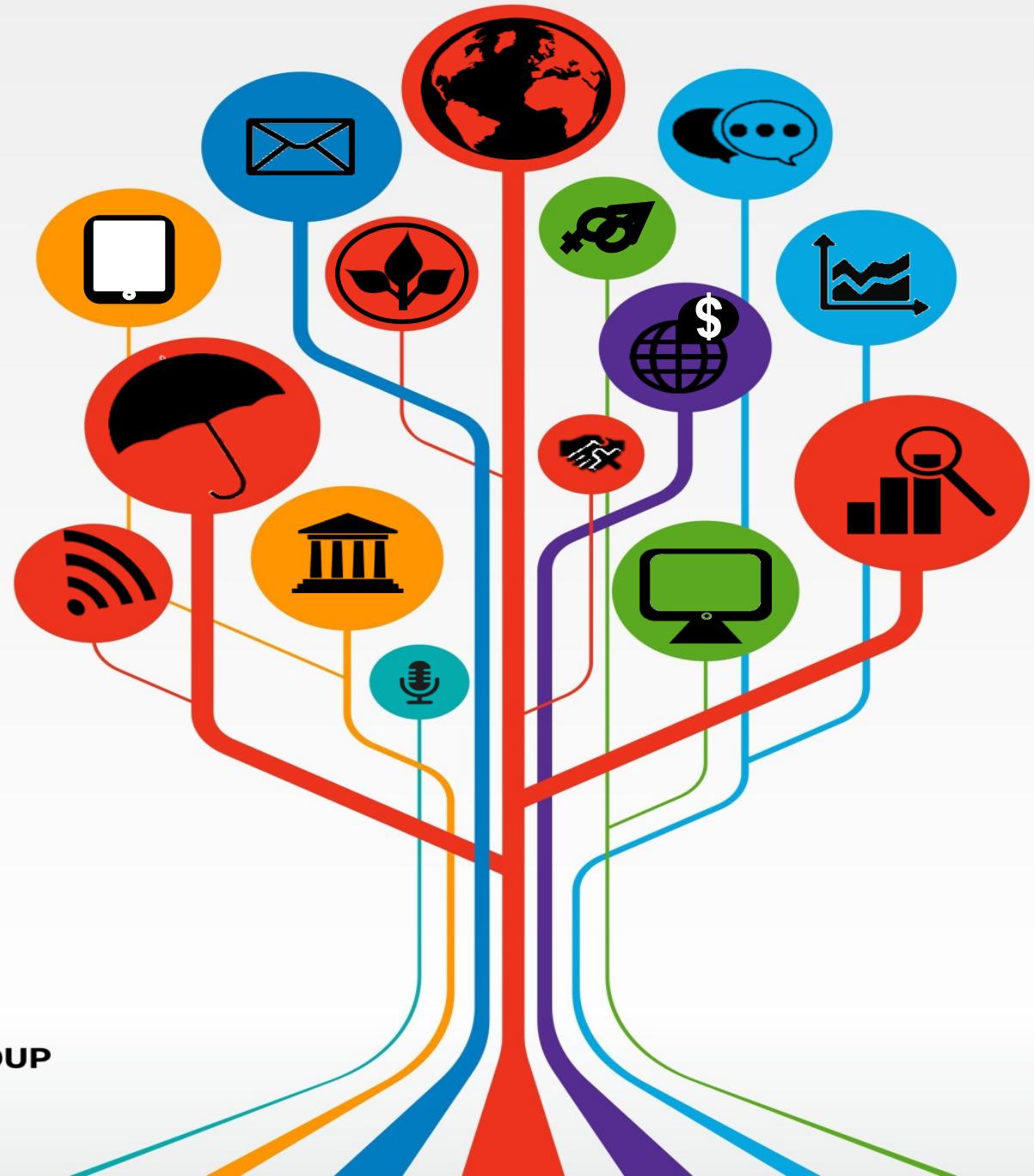
dimeanalytics@worldbank.org

Presented by Benjamin Daniels

bdaniels@worldbank.org

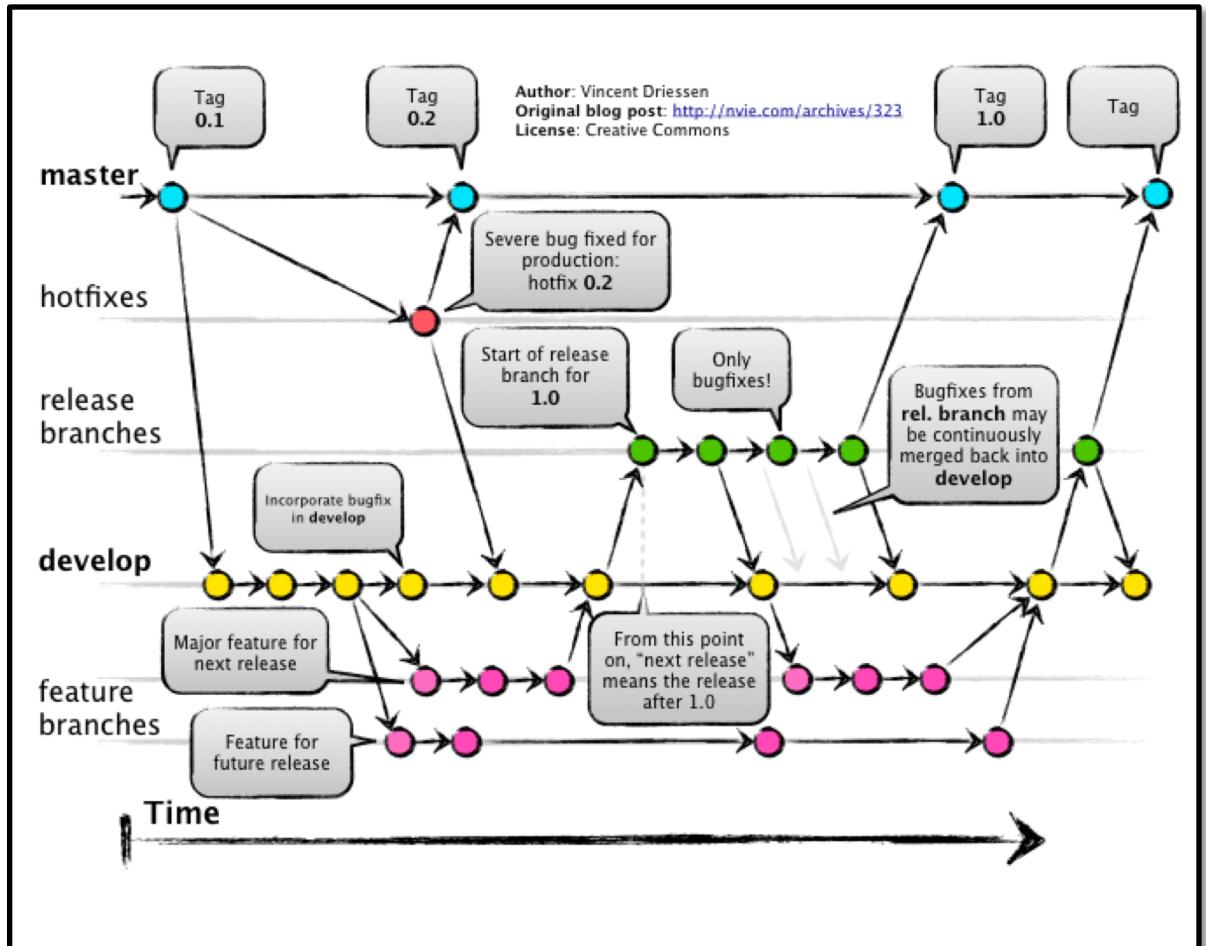


Git Flow: Essential Ideas



Git Flow is a branching model for Git

- Git Flow is what is called a “branching model”. It is a workflow for organizing Git branches and commits that makes sure people know what things are for.
- It is an industry standard model, so people spend no time “figuring out how to work”.



Wait, what's a “branch” and a “commit”?

- Git works fundamentally differently from other “version control” software (ie Dropbox)
- Dropbox stores sequential images every time you save
- You save often (right?)
- So Dropbox stores many similar images of your files
- And you can't really tell which is which

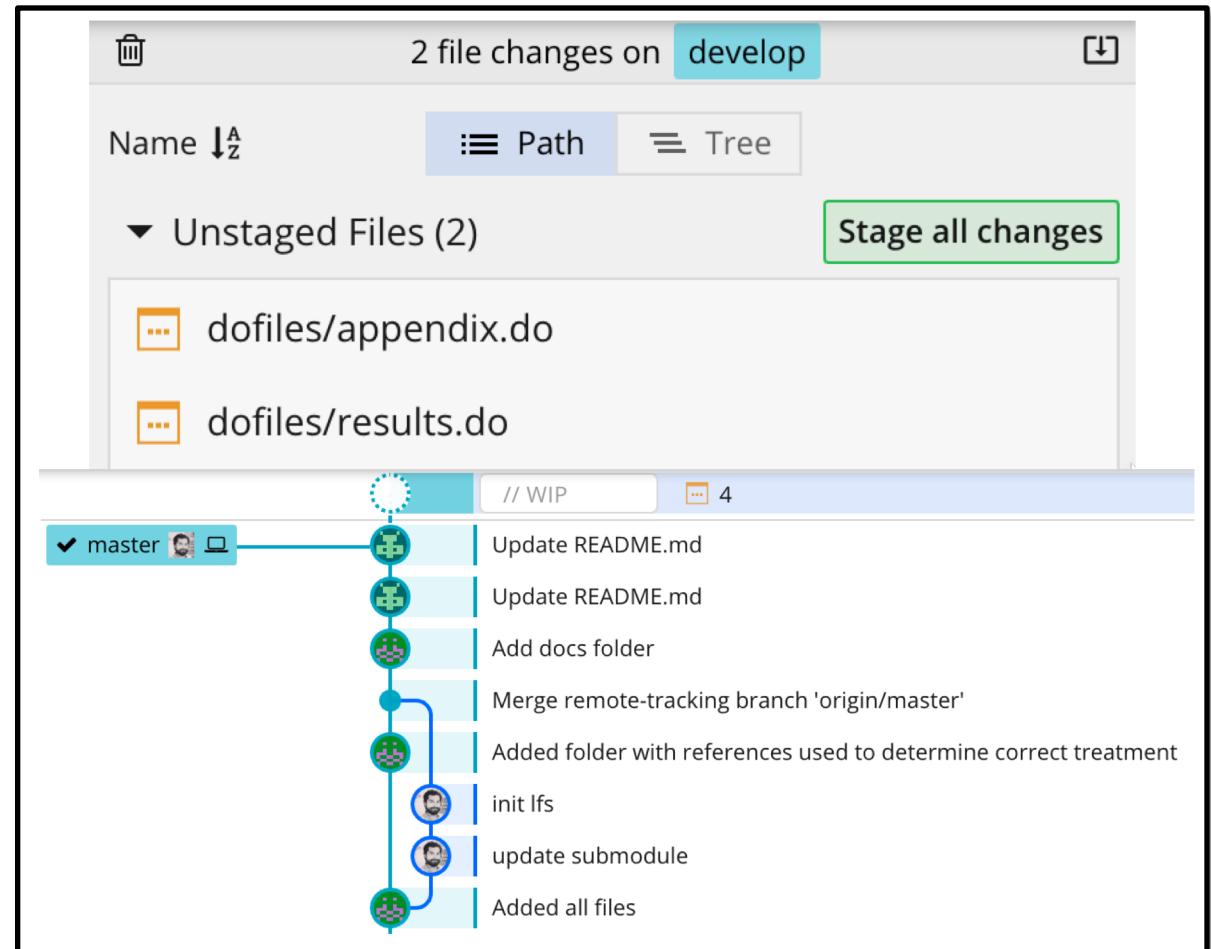
The screenshot shows a list of versions for the file 'baseline_results.do'. At the top, there is a search bar and a user profile icon. Below the header, a message states: 'You can restore any version below to make it the current file. All other versions will still be saved.' The first version listed is from 'August 6, 2018' at 5:09 PM, edited by Benjamin Daniels on Desktop, with a size of 4.99 KB. This is labeled as the 'Current version'. The second version is from 'June 12, 2018' at 3:39 PM, also edited by Benjamin Daniels on Desktop, with a size of 5.08 KB. A blue 'Restore' button is visible next to this entry. The third version is identical to the second, also from June 12, 2018 at 3:39 PM, edited by Benjamin Daniels on Desktop, with a size of 5.08 KB.

Date	Time	Editor	Location	Size	Action
August 6, 2018	5:09 PM	Edited by Benjamin Daniels.	Desktop	4.99 KB	Current version
June 12, 2018	3:39 PM	Edited by Benjamin Daniels.	Desktop	5.08 KB	
June 12, 2018	3:39 PM	Edited by Benjamin Daniels.	Desktop	5.08 KB	Restore

I can't believe how much I pay for this nonsense...

“Commits” give names to past versions

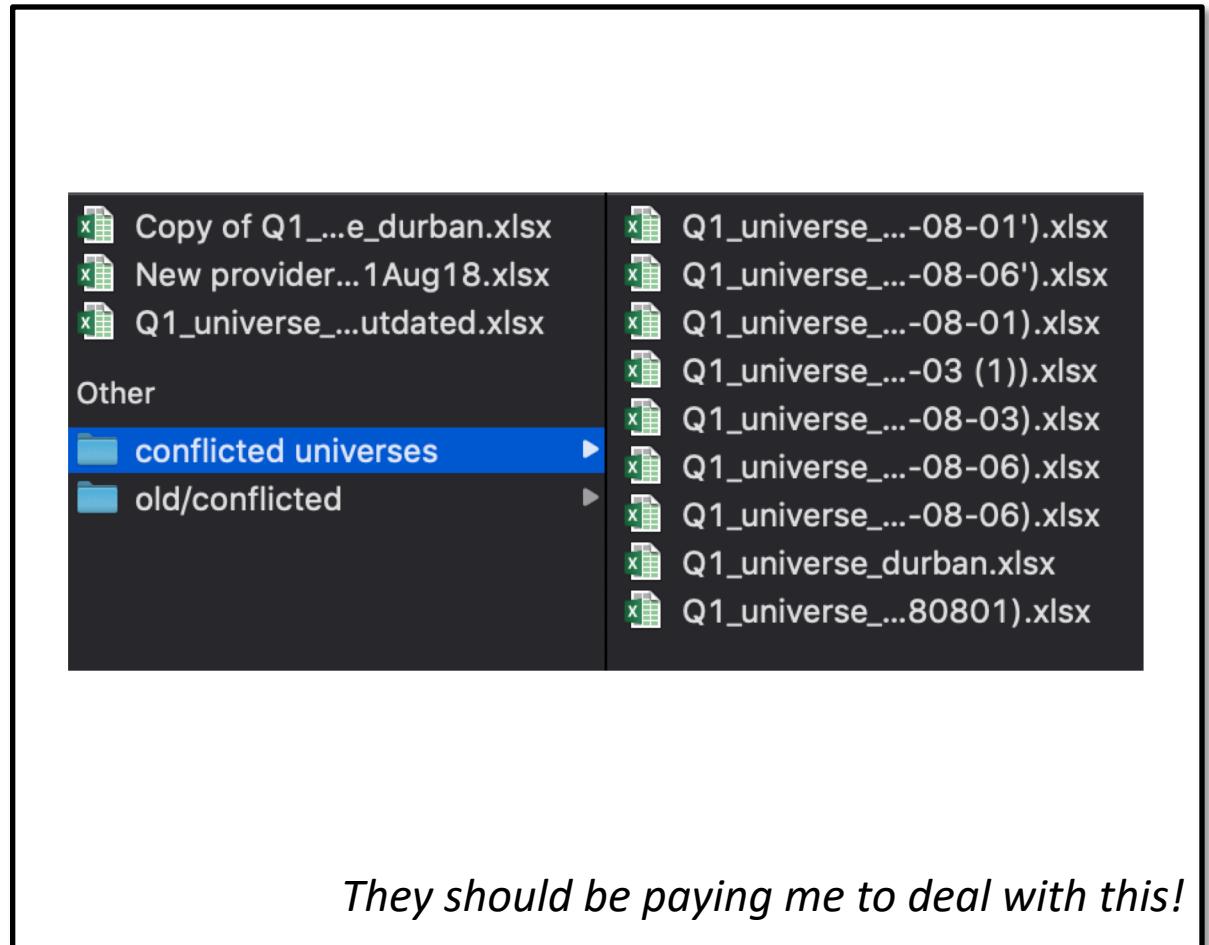
- Git *watches* your saved changes
- But it only *stores* them when you tell it to: this is a “commit”
- You can name these so that they make sense and are easy to find when you need them in the future (you *always* need them when you least expect, right?)
- Required Reading: Git vs Dropbox
<https://michaelstepner.com/blog/git-vs-dropbox>



THIS WEBSITE COSTS ZERO DOLLARS!

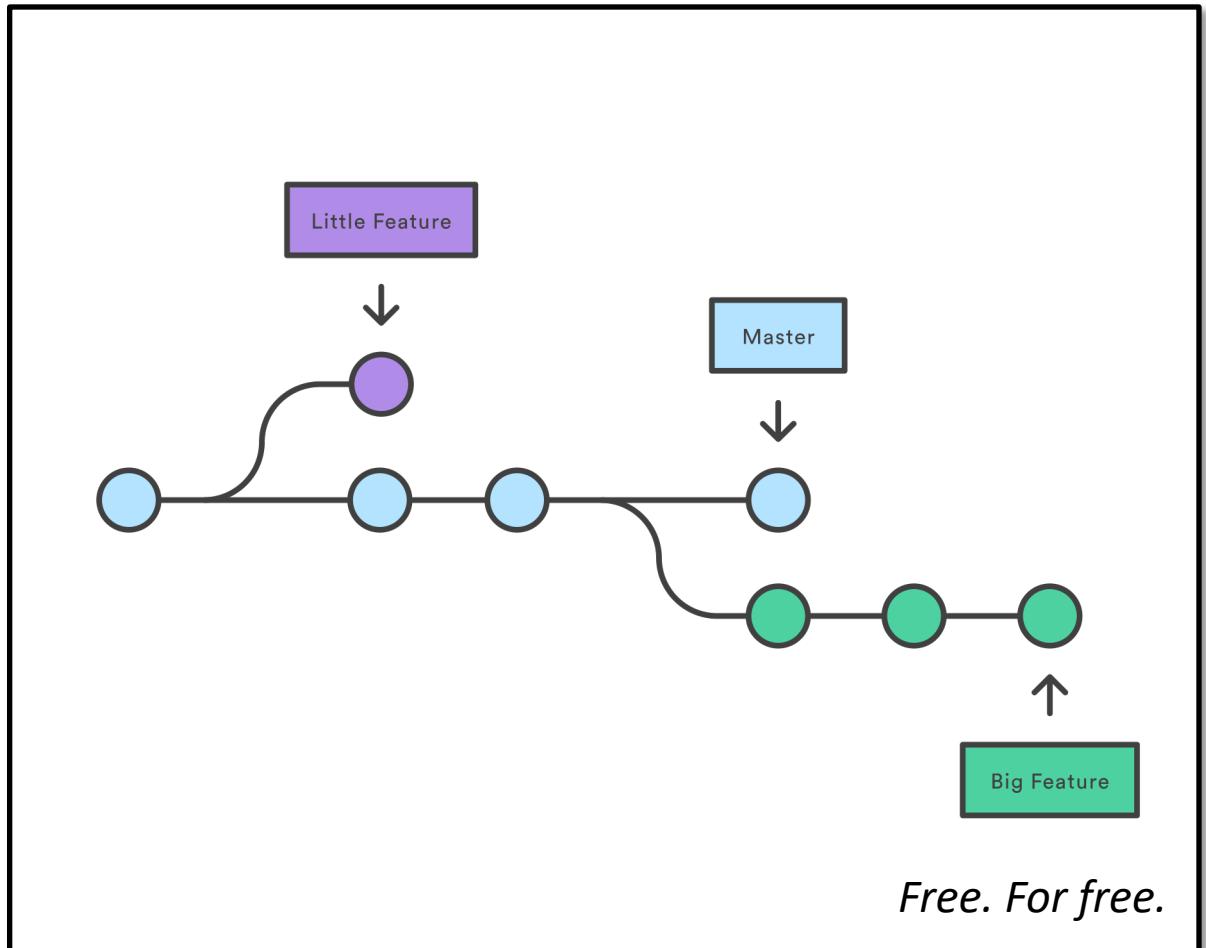
“Branches” give names to current versions

- Git also store simultaneous information differently than Dropbox.
- In Dropbox, there can be only *one* version of a file at any time.
- If people edit simultaneously, it's the stuff of nightmares
- Seriously, nightmares.



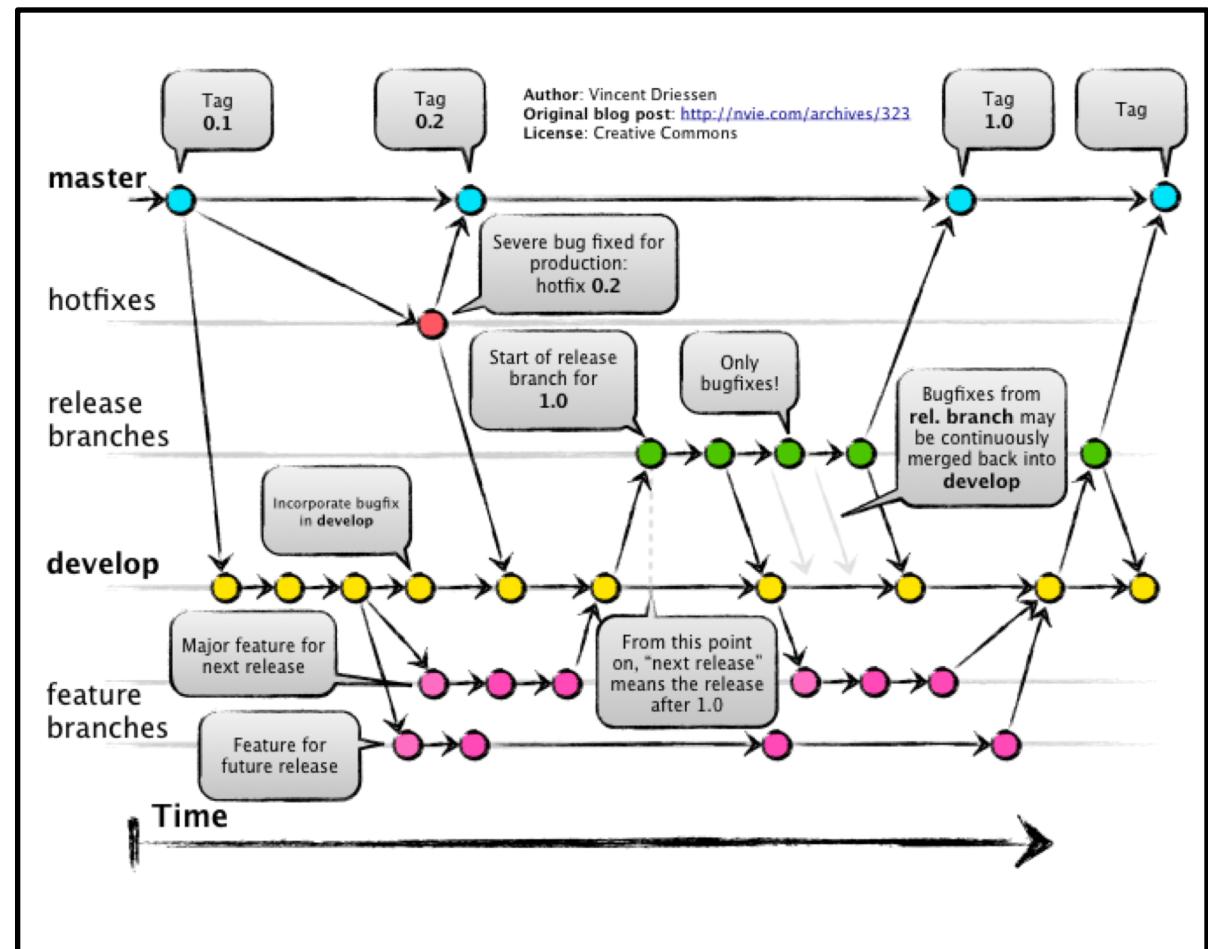
“Branches” give names to current versions

- Git lets you get around this problem as an extension of its “commit” feature
- Since you can “name” your history snapshots, you can also put them in folder-like structures called “branches” – for series of changes that make sense together
- These all exist simultaneously and are effortless to switch between – we’ll practice soon.



Git Flow organizes your commits and branches

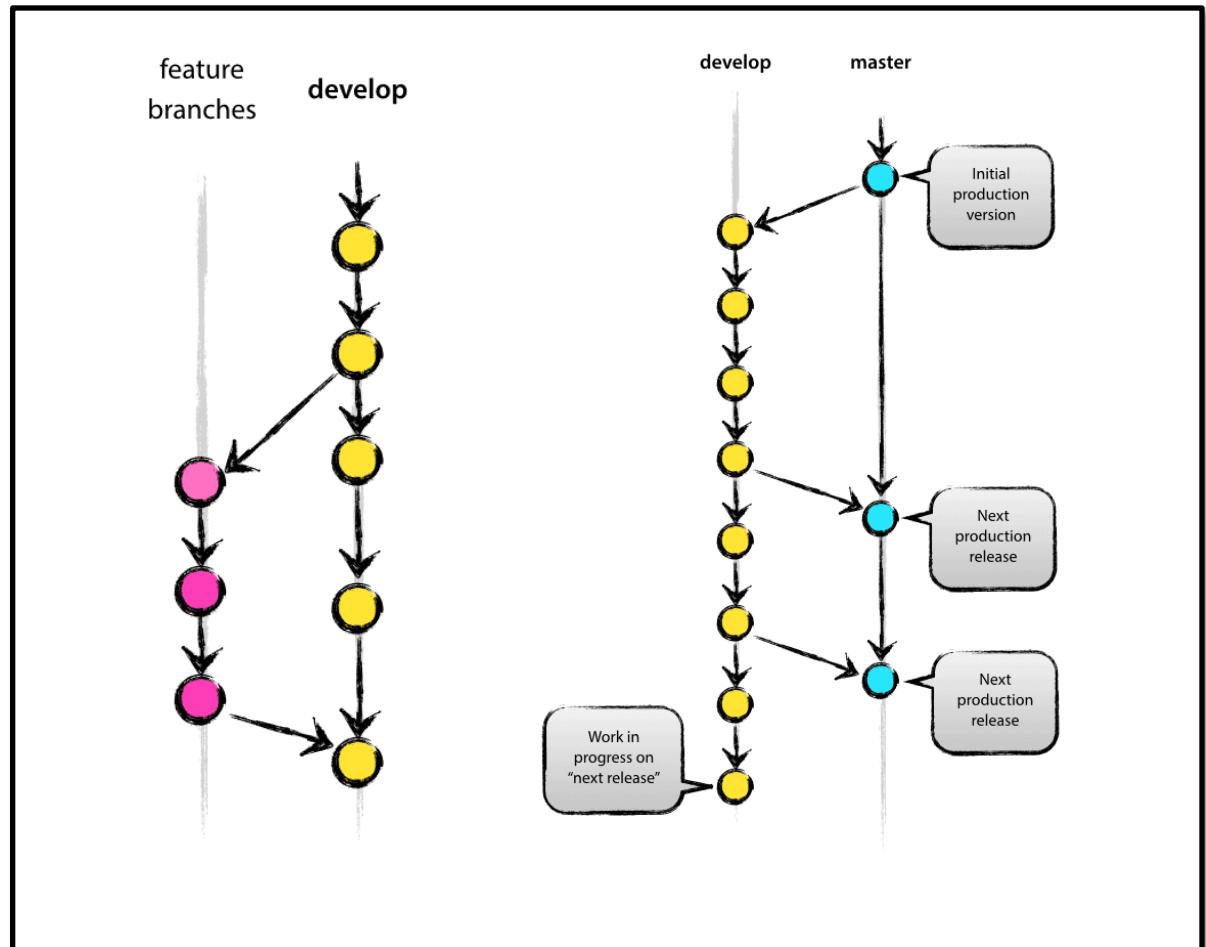
- Git Flow is a similar idea to the standardized [*iefolder*] structure
- When you move to another project using Git Flow, you will know what everything is for and how you should be working
- Required Reading: Git Flow
<https://nvie.com/posts/a-successful-git-branching-model/>



Why Git Flow?

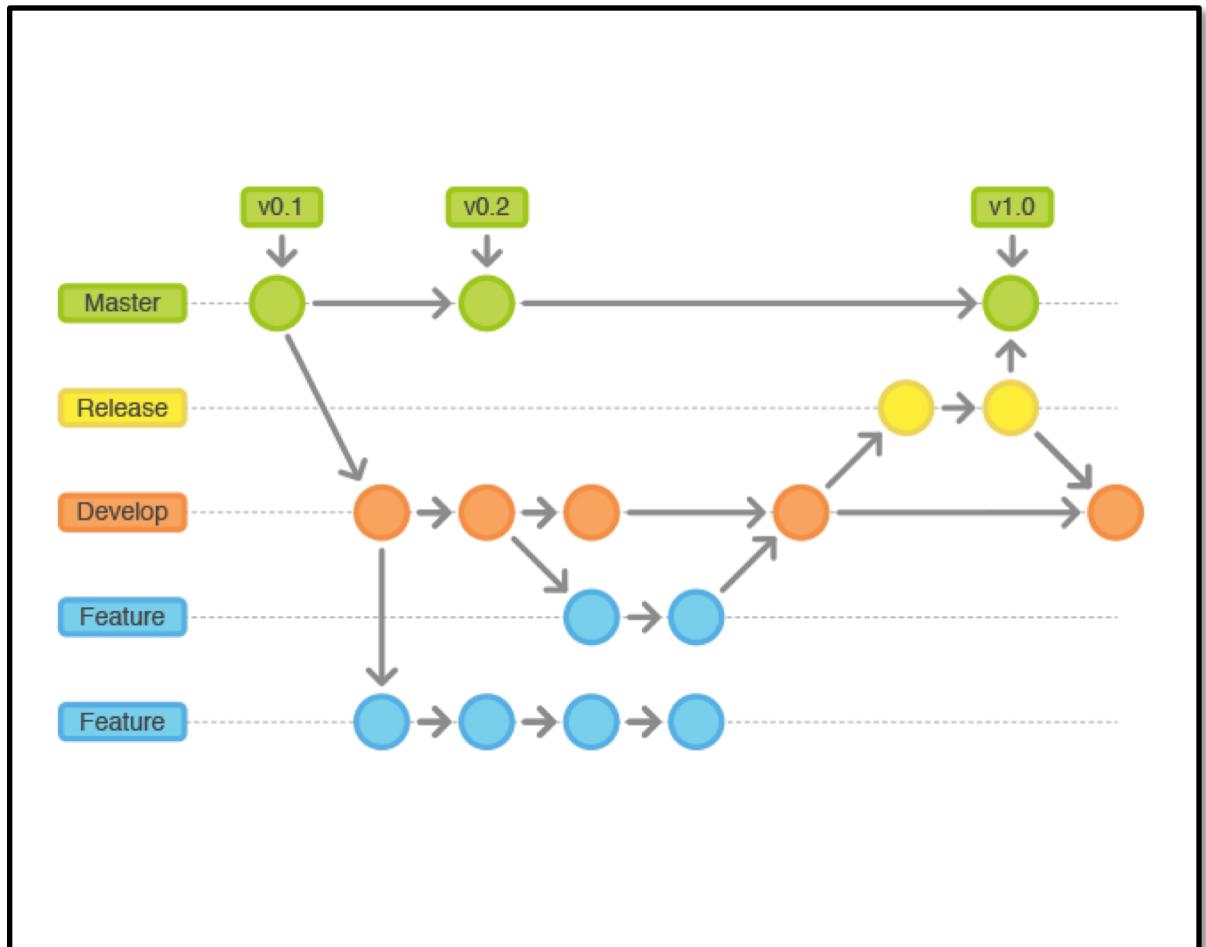
Three core ideas:

1. *[master]* branch is protected, always ready to run, and has sparse commits from *[develop]*
2. *[develop]* branch is protected, always ready to run, and has frequent commits from *[feature]*
3. *[feature]* branches are frequent, specific, and where all new work is done

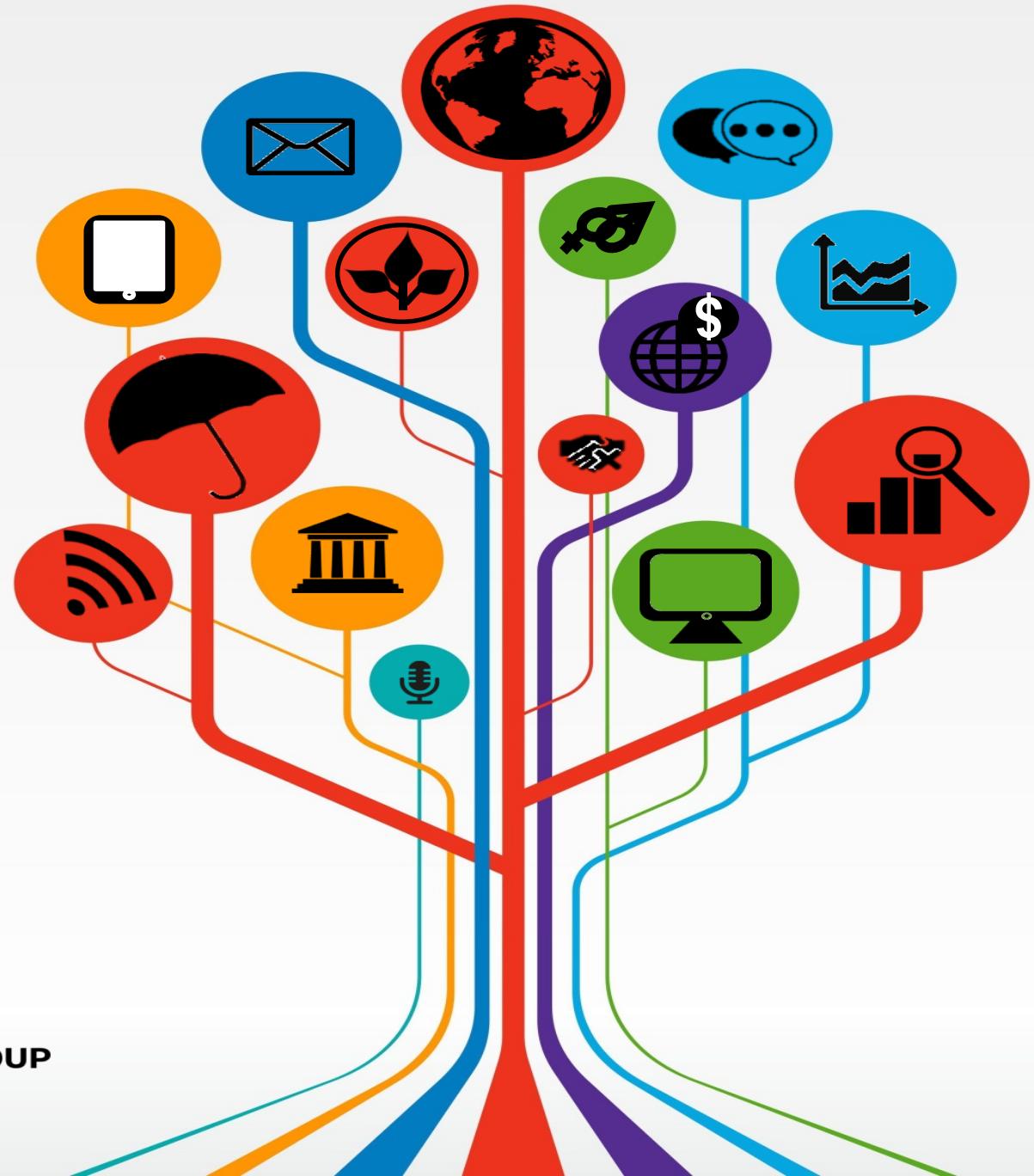


Benefits of Git Flow

- Anyone (like the PI) can *always* open [*master*] branch and see “where we are”.
- Anyone (like the PI) can create a [*feature*] branch and ask “what happens if I do X?”
- You can always look back and see what happened when someone tried something else (abandoned [*feature*] branches)

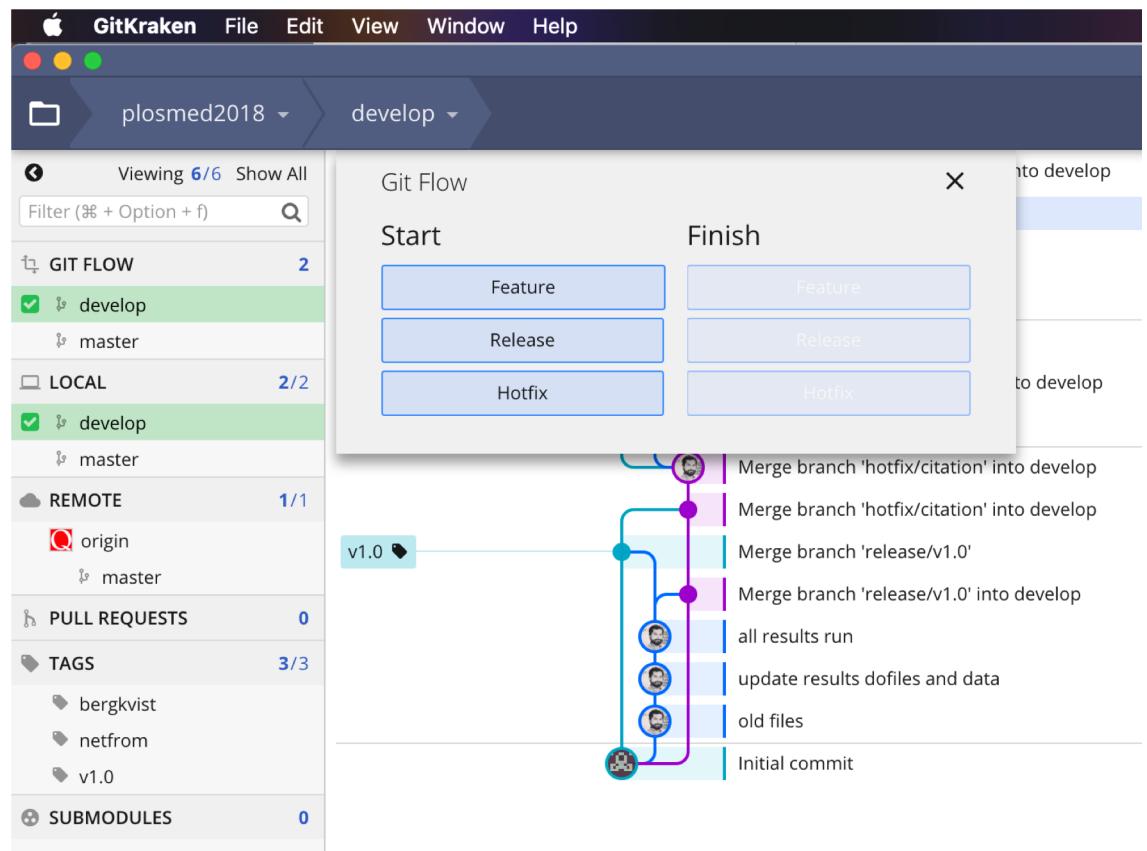


Git Flow: Practical Example



Using Git Flow with GitKraken

- Git Flow is so successful it is the default in many organizations, and GitKraken has powerful automatic features for supporting Git Flow.
- [Install GitKraken](#) and log in using your [GitHub account](#).
- Send me your GitHub username so I can give you edit access on the git-flow-demo repo



Clone the git-flow-demo repo locally

Repository Management X

Open	Clone with URL
Clone	GitHub.com GitHub Enterprise GitLab.com GitLab (self-hosted) Bitbucket.org Visual Studio Team Services
Init	

Clone a Repo

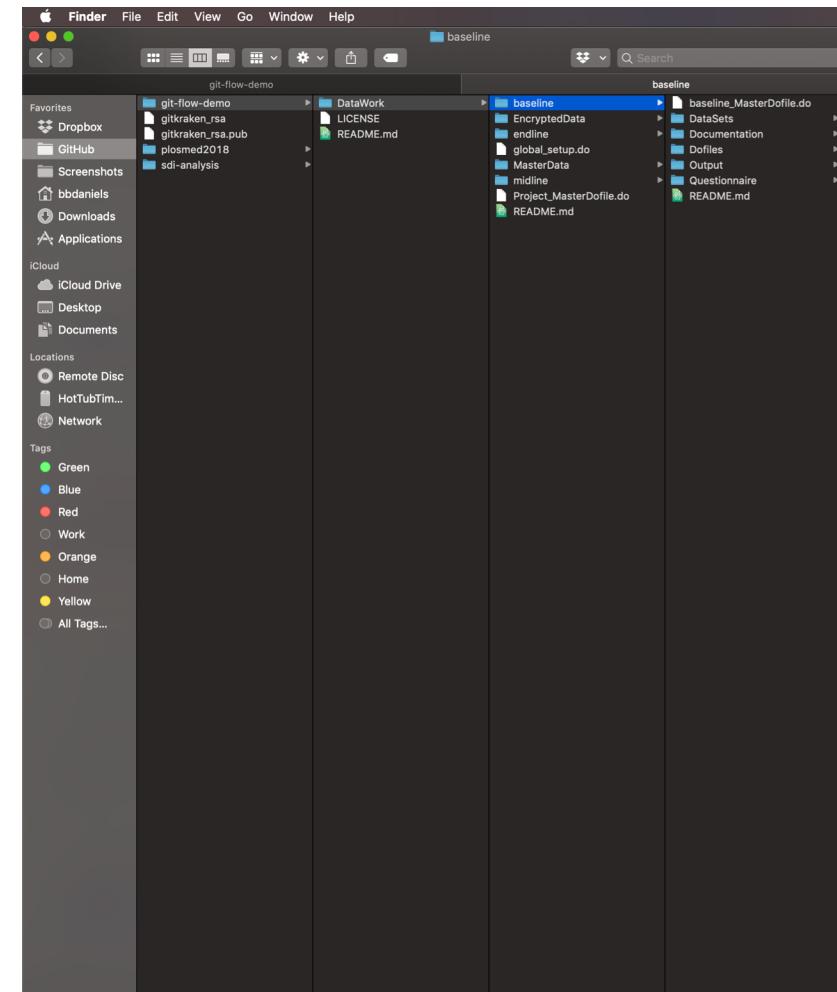
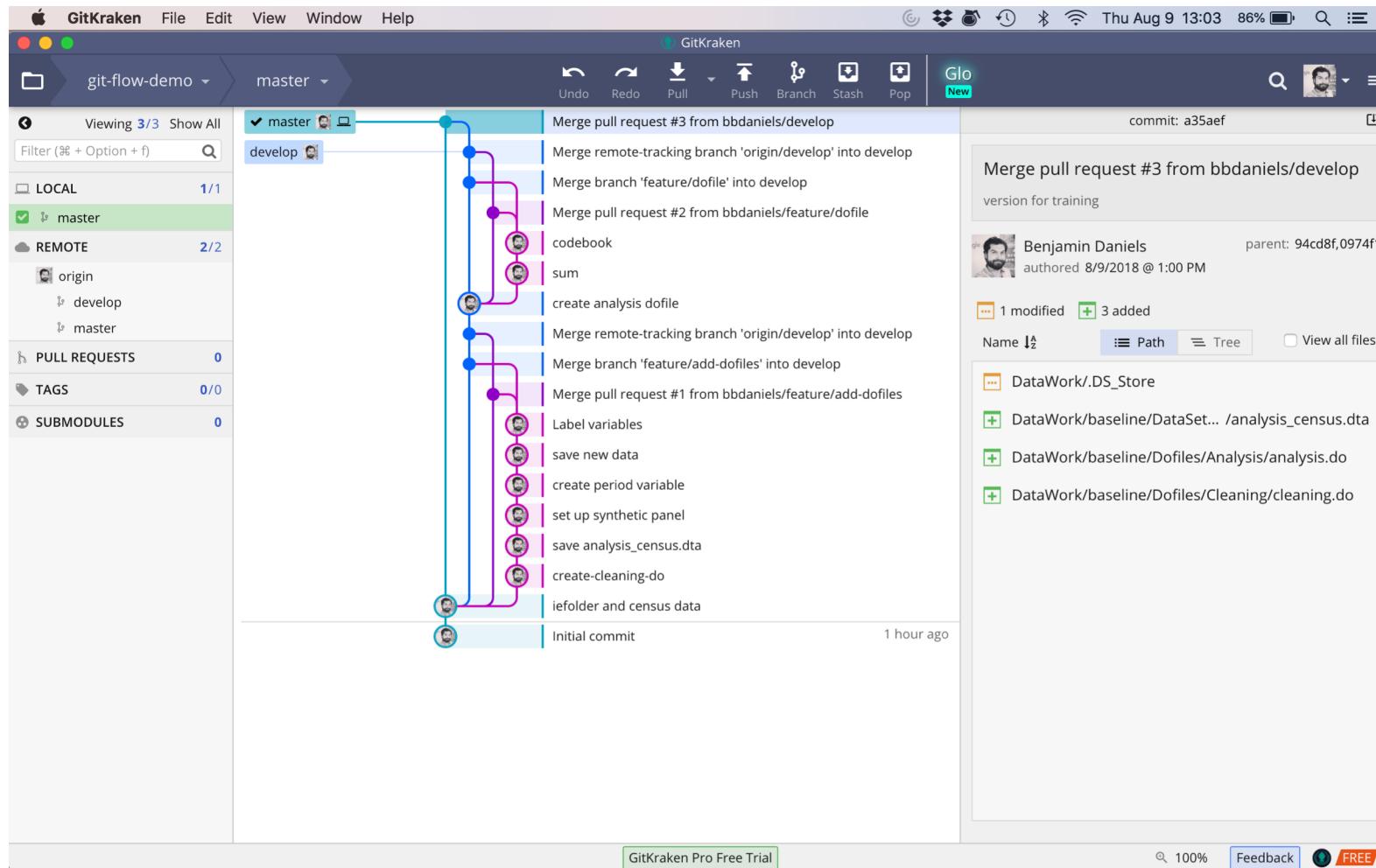
Where to clone to Browse

URL

Full path git-flow-demo

Clone the repo!

Open the git-flow-demo repo and folder



Initialize Git Flow in “Preferences”

Exit Preferences

Default Profile
Benjamin Daniels
bbdaniels@gmail.com

General

Profiles

Authentication

UI Preferences

Repo-Specific Preferences

git-flow-demo
/Users/bbdaniels/GitHub/git-flow-demo/

Git Flow

Git Flow

Branches

Master	master
Develop	develop

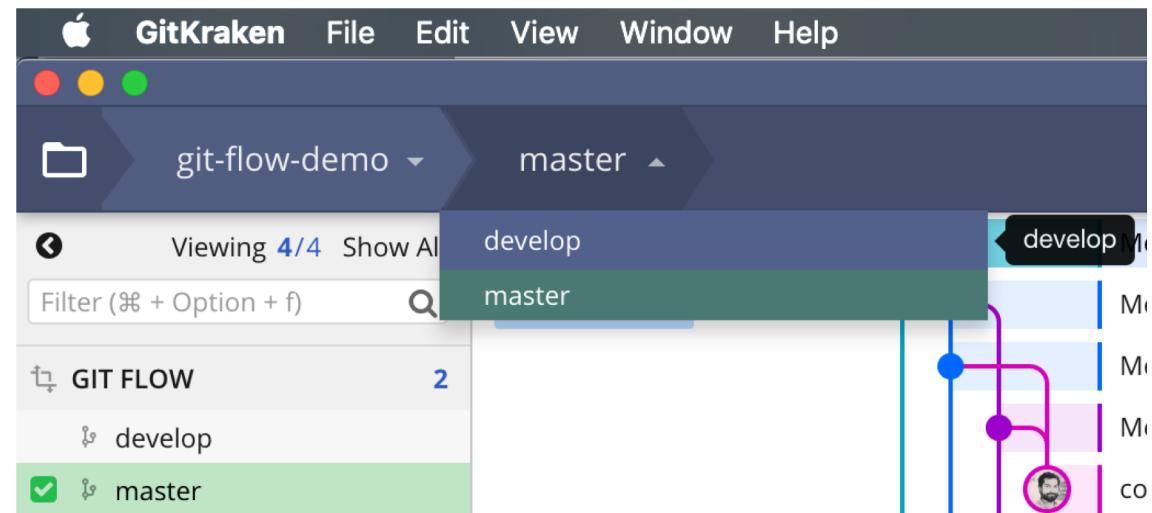
Prefixes

Feature	feature/
Release	release/
Hotfix	hotfix/
Version Tag	

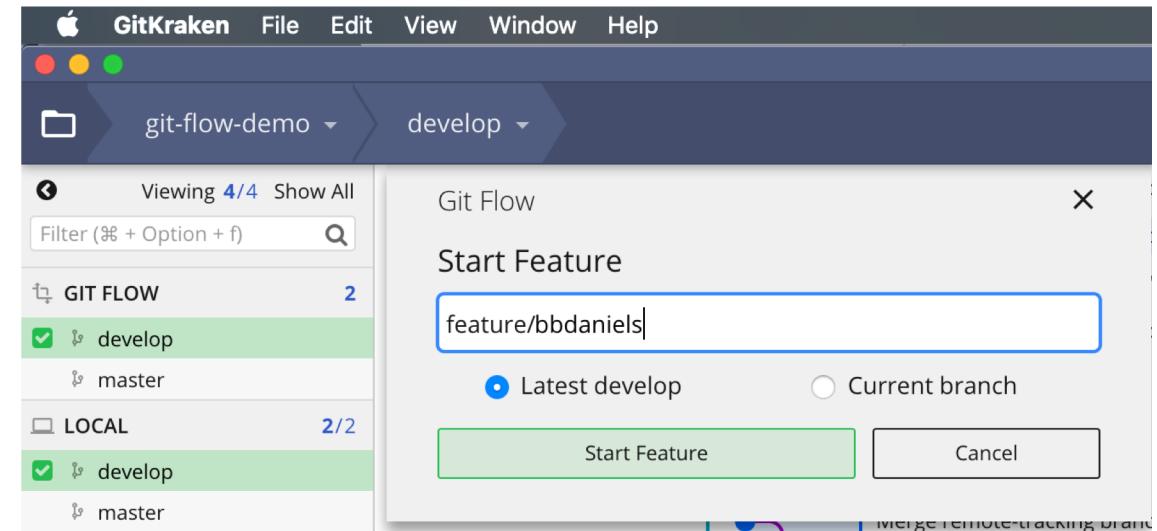
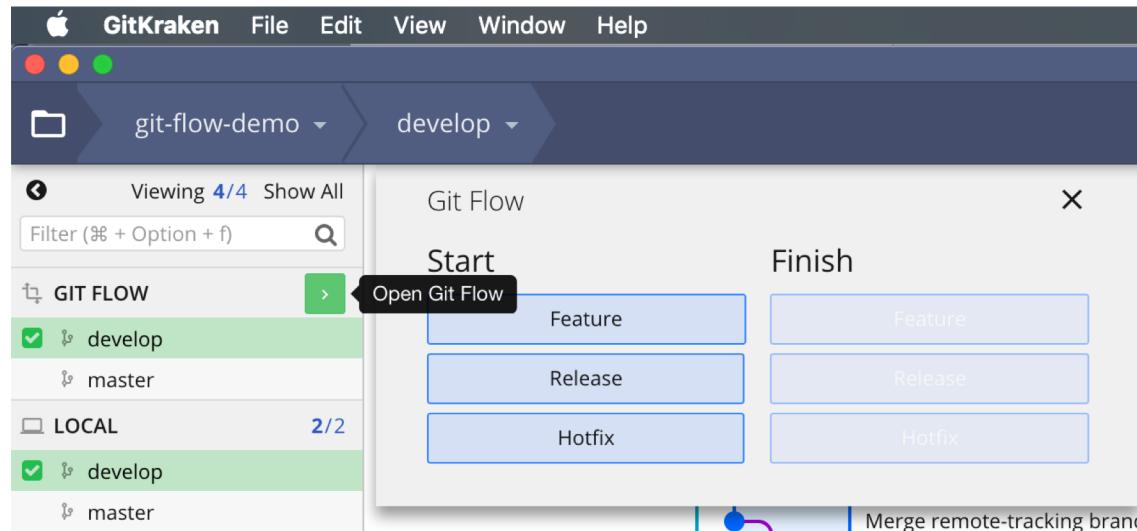
Initialize Git Flow

Checkout “develop”

- When you “check out” a branch, everything on your computer is instantly updated to what that branch looks like.
- Always be conscious to remember “where” you are – this is the core functionality of Git. You’ll get used to it!

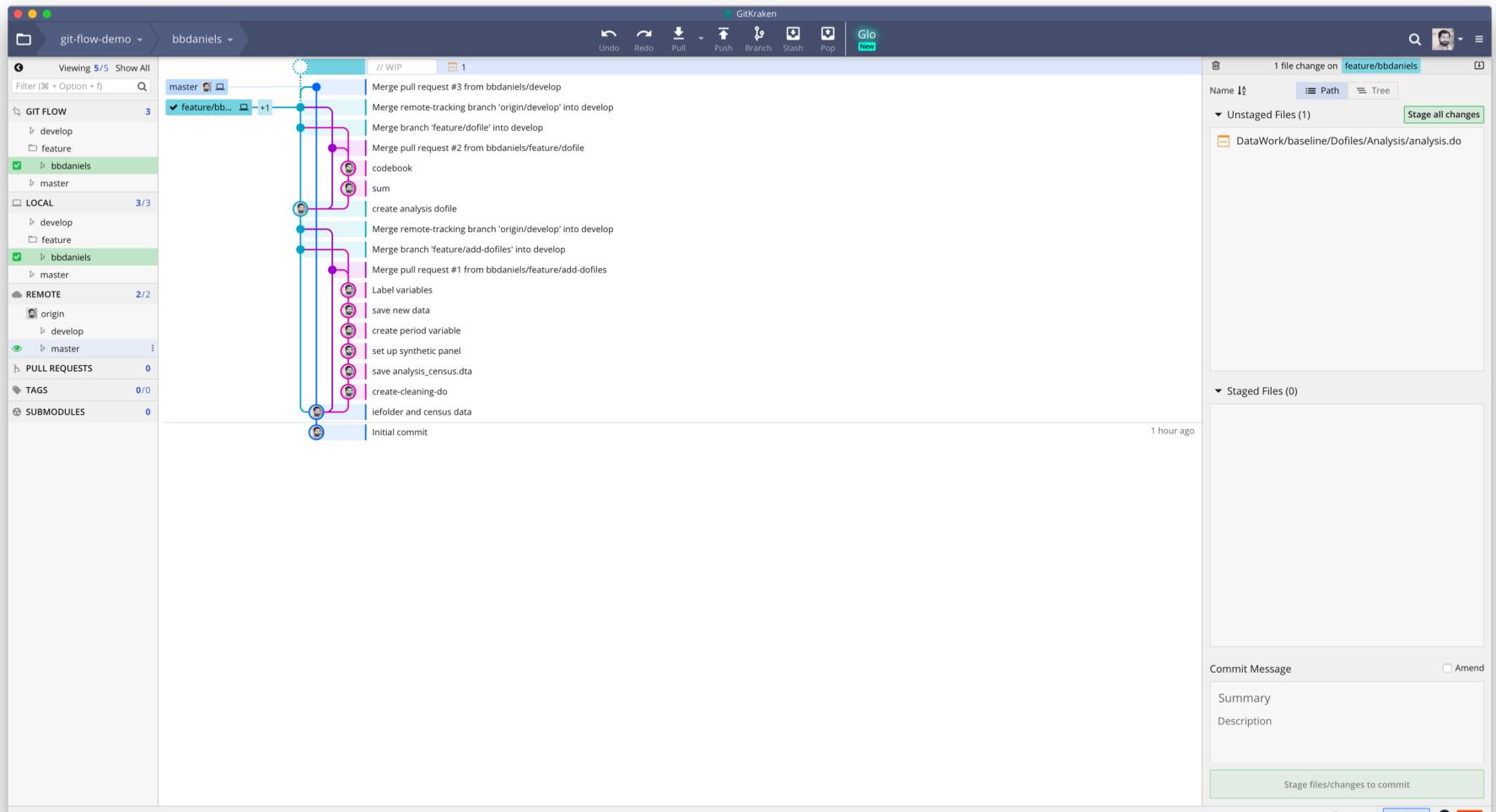


Use Git Flow to start a new feature

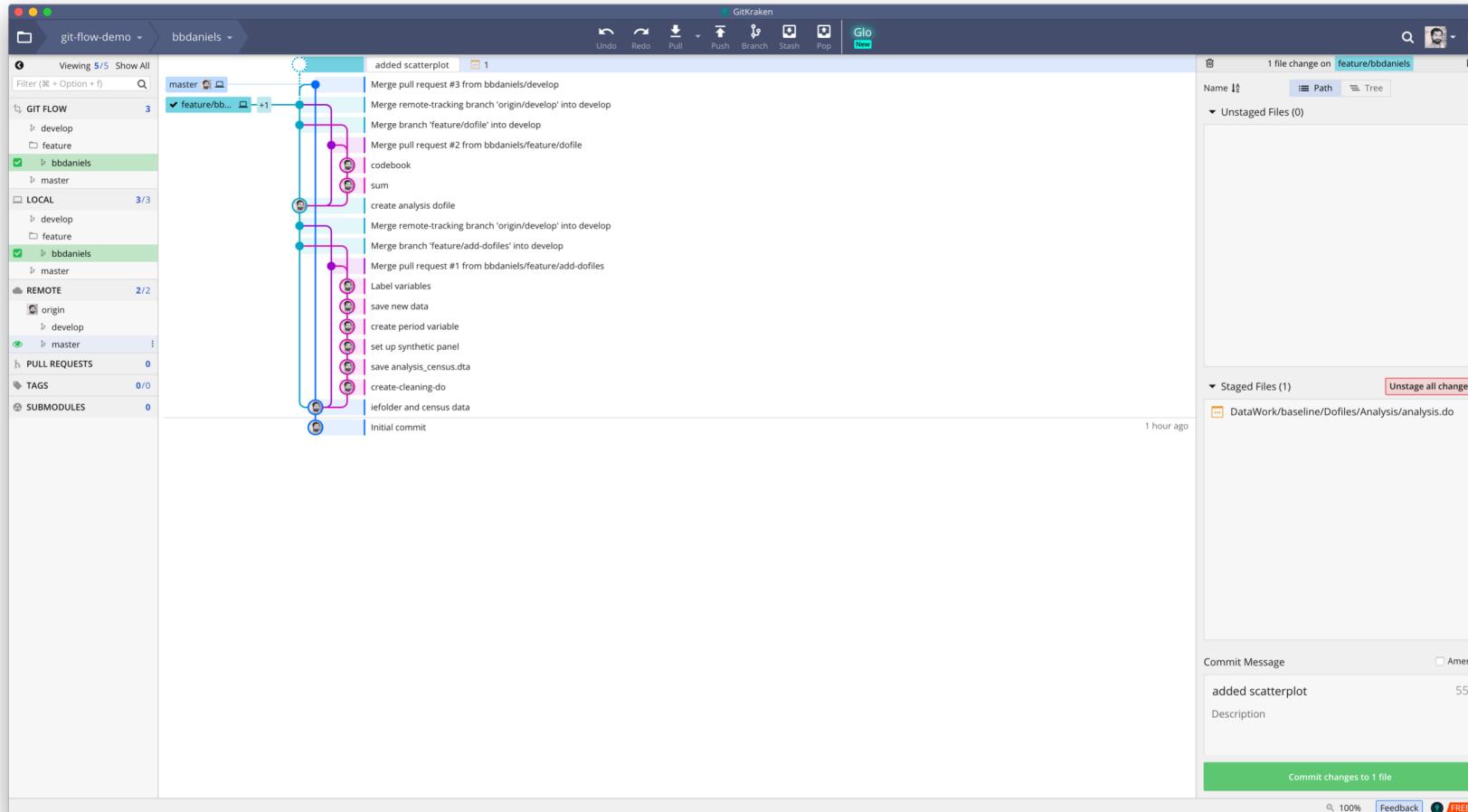


Open the analysis dofile and do something!

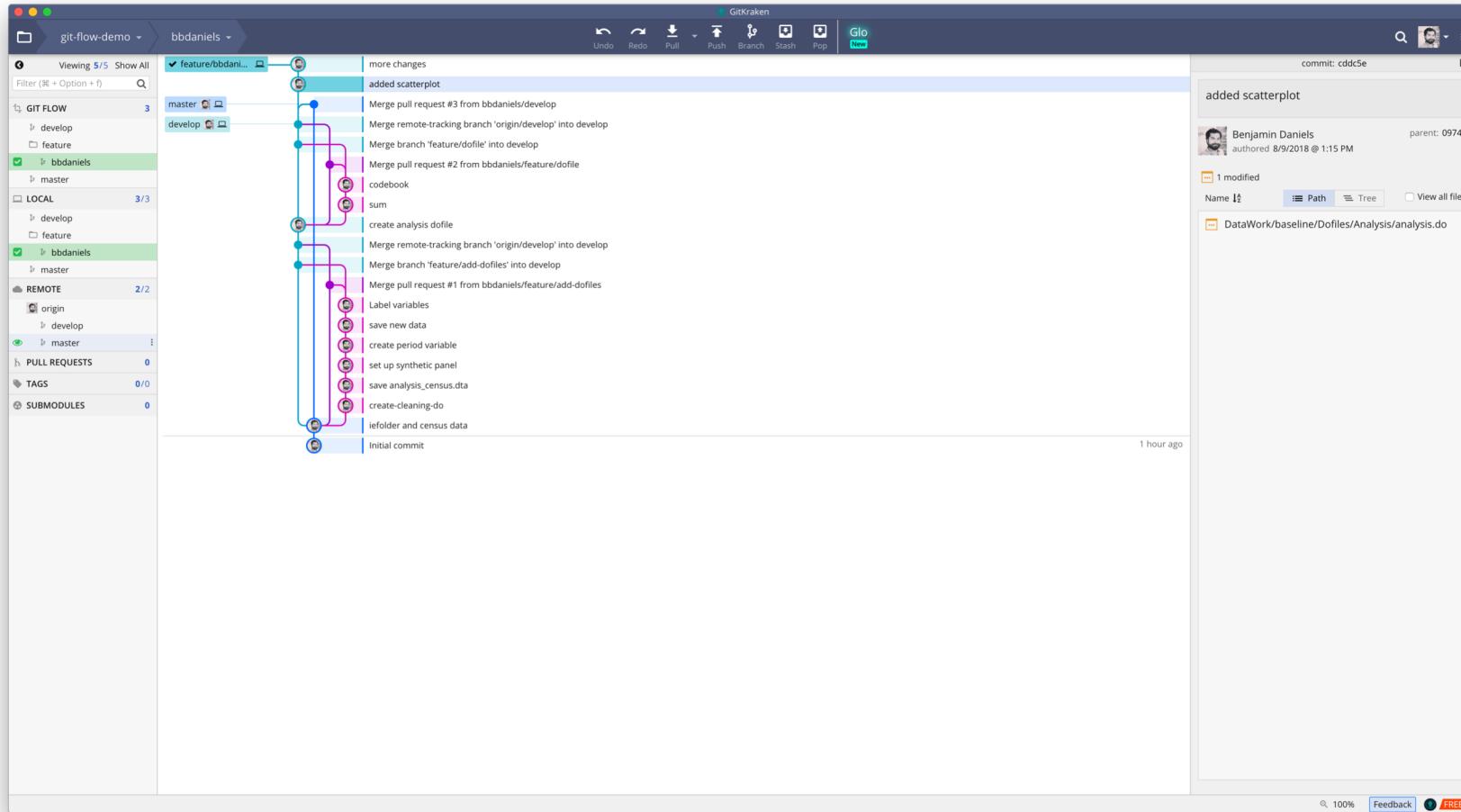
- Remember you will need the Baseline Master open and run to have the correct \${globals} in place to run this.
- Once you've edited the dofile and it runs, save it.
- This puts “unstaged changes” in your WIP (“work in progress”).



Stage and commit your changes

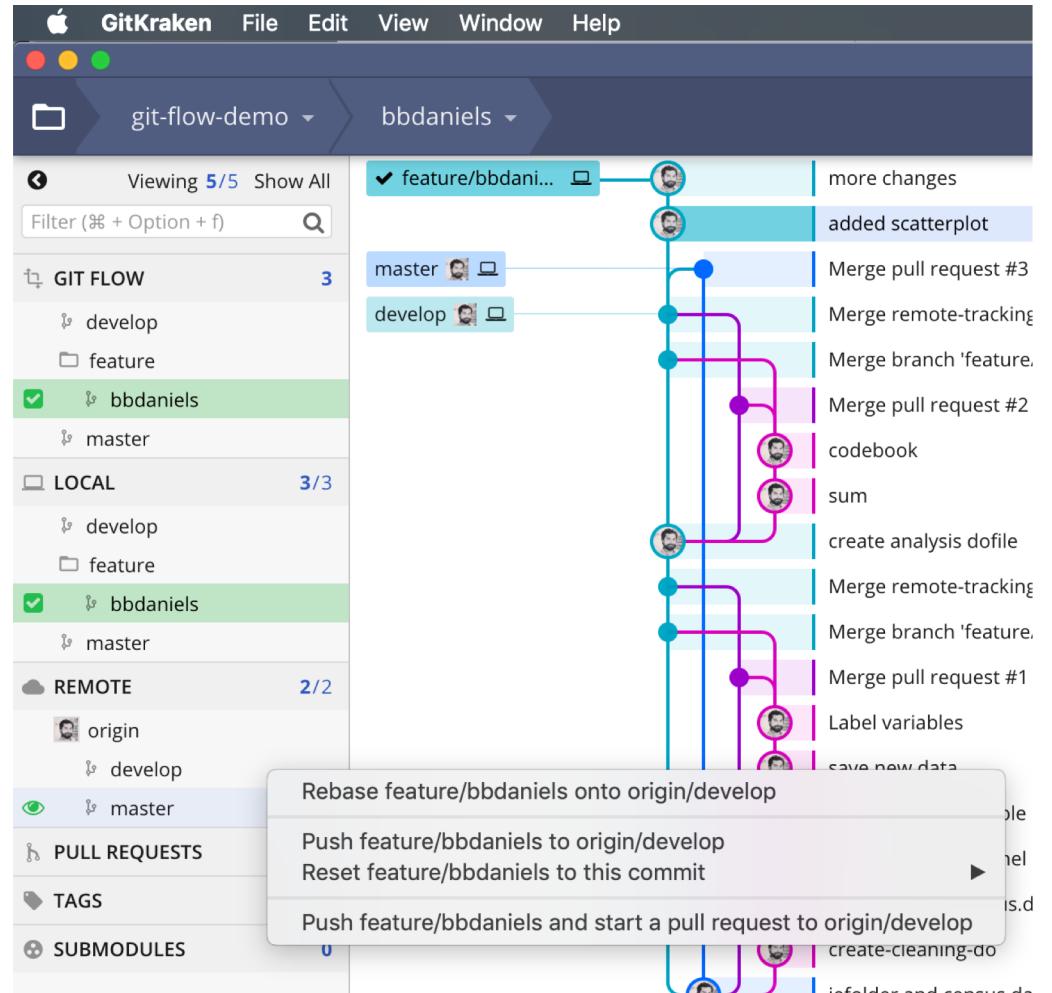


Change something else and commit again

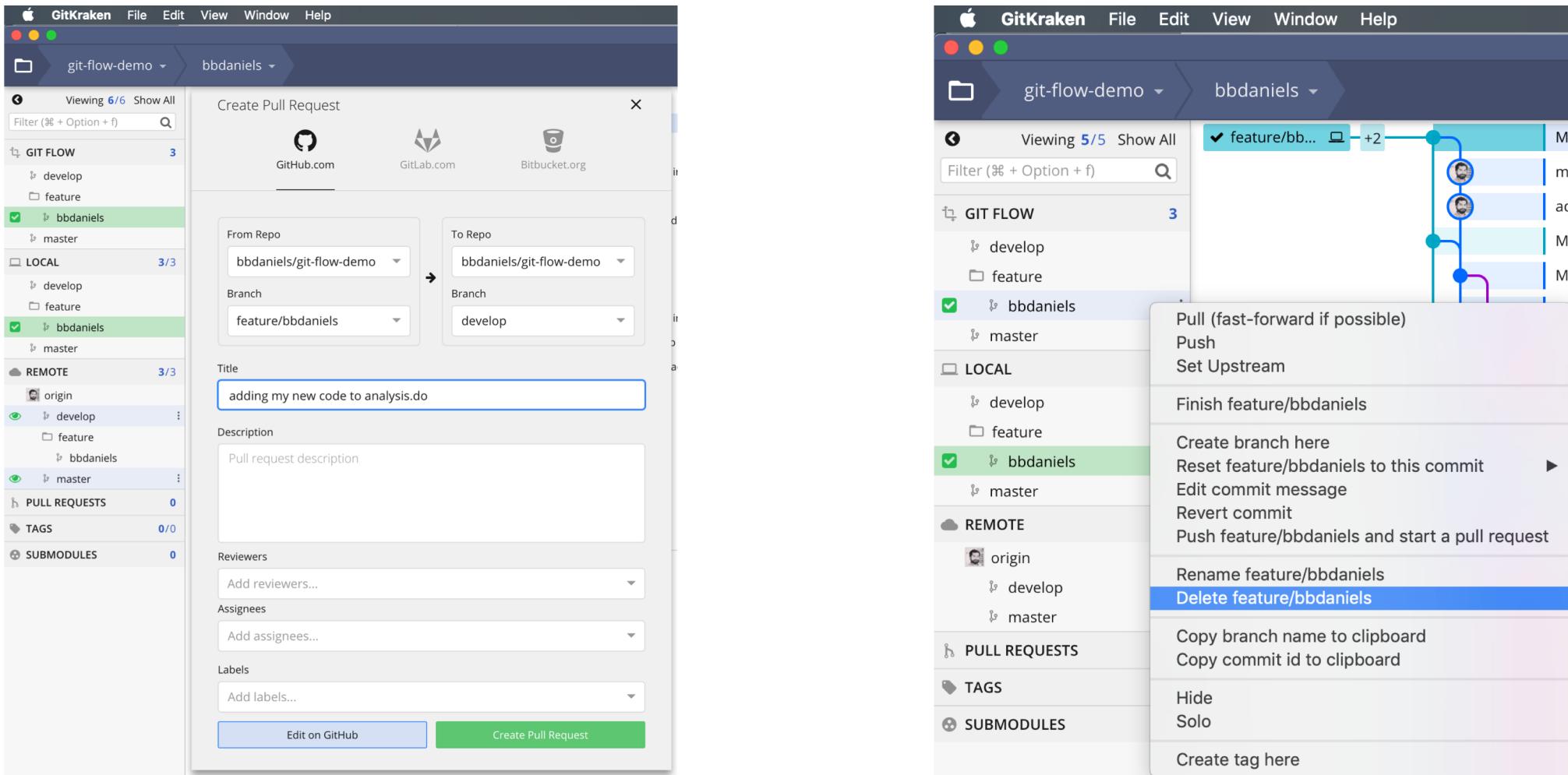


Push your branch and start a PR to origin/develop

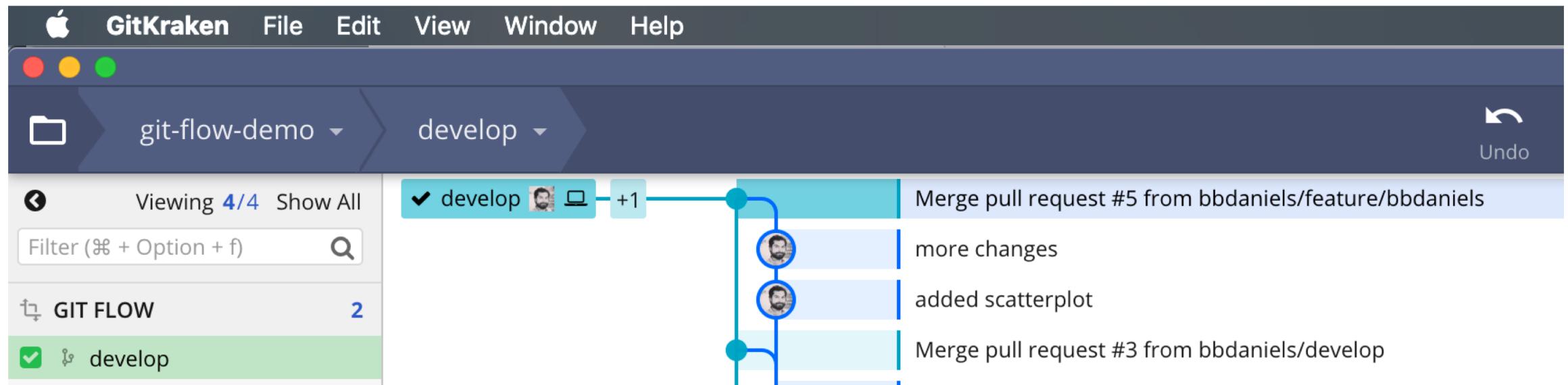
- Drag your branch from Git Flow to origin/develop
- Select the last option



Create the Pull Request and delete your branch locally!

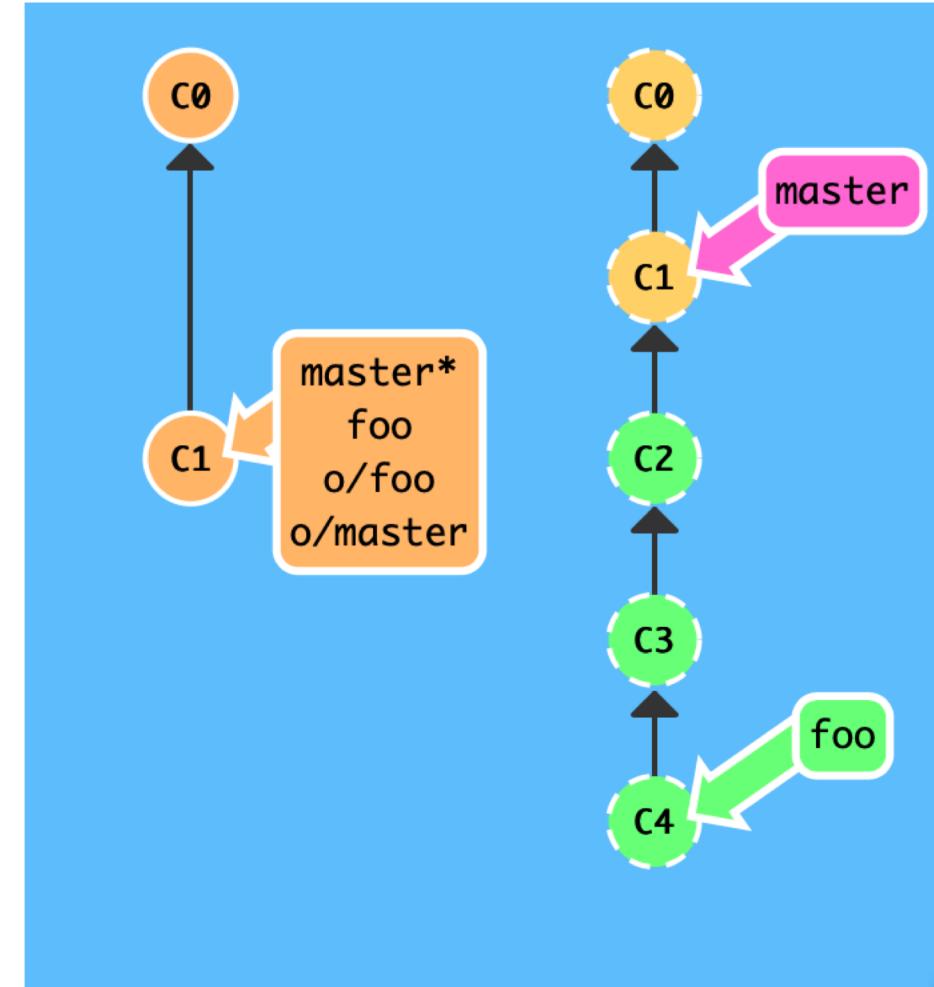


Next: I'll merge these on GitHub



Homework: play with toys

- Practice Git branching at:
<https://learngitbranching.js.org>
- Type [levels]
- Complete Level 1-4 of
“Introduction Sequence” on the
Main tab
- Complete all levels on the
Remote tab



Combining GitHub and Dropbox

- You will probably still want to store certain files on Dropbox – namely, data.
- However, GitHub and Dropbox don't mix intuitively, since they sync in very different ways.
- But this is no problem at all.
- Data can move instantly into the repo with [exportCodebook](#).
- Other options:
[https://github.com/kbjarkefur/GitH
ubDropBox](https://github.com/kbjarkefur/GitHubDropBox)

Thank you!

