



WHY GIT & GITHUB?

Why would Git/GitHub interest me? An introduction.

James R. Small, Sr. Infrastructure Architect

Michigan! /usr/group

mug.org – A Free and Open Source Michigan Community

YET ANOTHER TOOL??? WHY GIT/GITHUB?

- I have no time and too many tools
- Why would I even consider looking at Git/GitHub?
- What problems do these tools solve?

COMMON PROBLEMS

- I changed (a) program/script/configuration/markup/markdown file(s) and something broke – how do I revert to the last working version?
- I need to test a new feature/option without breaking the main/production version
- We have a team working on code/configuration/markup and need to know who did what

PROBLEMS SOLVED BY GIT

Git can:

- Lets me easily version or “checkpoint” files
 - » Each checkpoint time and date stamped
 - » Includes comments on what changed
 - » Allows diff between versions to see exact changes
 - » Allows rolling back to previous versions

PROBLEMS SOLVED BY GIT

Git allows branching:

- Lets me easily duplicate my entire file based setup as a new branch
 - » I can easily switch back and forth between branches (e.g., production and test)
 - » I can easily try things out without breaking production
 - » If I like something I can easily merge it back in, if not I can easily discard

PROBLEMS SOLVED BY GITHUB

GitHub is a shared storage for Git:

- Allows a team of people to easily collaborate on a project
 - » I can easily see who did what
 - » I can track feature requests/bugs
 - » I can integrate automatic testing – every time someone checks in a change it will be tested with the results shown (Continuous Integration)

WHY GIT/GITHUB OVER OTHER SOLUTIONS

Git/GitHub may be interesting,
but why these tools over other
similar ones?

WHO USES GIT/GITHUB?

- the Linux Kernel
- Ruby on Rails
- Bootstrap
- Node.js
- JQuery
- D3
- Homebrew
- vim!!!
- [GitHub Repository Ranking](#)

WHITHER GIT/GITHUB?

- Managing large projects – code, web sites, infrastructure – with a (potentially large) team is not easy
- Distributed Source/Version Control Systems tackle this challenge
- In 2005 the best option (according to Linux Torvalds) was BitKeeper but it was proprietary and had some limitations

WHITHER GIT/GITHUB?

- A spat between the Linux Kernel Developers and BitKeeper resulted in Mr. Torvalds creating a new system from scratch – git – after all, how hard could it be? 😊
- Torvalds quipped about the name *git* (which means "unpleasant person" in British English slang): "I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'."

WHITHER GIT/GITHUB?

- A few years after Git's creation, GitHub came along
- Essentially a centralized cloud repository (shared storage) for git projects with a nice Web front end
- Also adds issue tracking, task management, wikis and is free for public projects
- The largest host of source code in the world

OBTAINING GIT

- Download the free client
 - » git-scm.com/downloads (CLI)



- » git-scm.com/download/guis (GUI)

GIT DOCUMENTATION

- git-scm.com/docs (git reference page)
- [git pdf cheat sheet](#)
- [git visualization cheat sheet](#)
- git-scm.com/book (Pro Git 2nd ed, free!)

GIT – GETTING STARTED (LOCAL)

- Start locally

- » mkdir test
- » cd test
- » # Initialize git – one time deal
- » git config --global user.name "James R. Small"
- » git config --global user.email james.r.small@example.com
- » # Check config: git config -l
- » # Start a local repository
- » # (directory with versioning/checkpointing)
- » git init
- » # Creates .git directory

GIT – GETTING STARTED (LOCAL)

- Local Repository (Repo)

- » # In test directory where .git directory is
- » # Create new file to be versioned:
- » vim notes.md

```
# Using git
* git is a powerful version control system
* git lets you use local repos
* git lets you version/checkpoint files to make it easy to track revisions

# Personalizing git
* Don't forget to configure your username and E-mail
git config --global user.name "your_username"
git config --global user.email "your_email@domain.com"
```

- » # What's the status of the file? Check at any time:
- » git status

```
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        notes.md

nothing added to commit but untracked files present (use "git add" to track)
```

GIT – GETTING STARTED (LOCAL)

- Local Repository (Repo)

- » # Check in (version) file with git add:
- » git add notes.md
- » # File is now staged – status:
- » git status

```
on branch master
Initial commit
changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   notes.md
```

- » #Commit changes to local repo:
 - » git commit -m "Initial commit"
- [master (root-commit) 7b23067] Initial commit of my notes
1 file changed, 13 insertions(+)
create mode 100644 notes.md

GIT – GETTING STARTED (LOCAL)

- Local Repository (Repo)

- » # Current status:

- » git status

```
jim@ubuntu16s1:~/test$ git status
On branch master
nothing to commit, working directory clean
```

- » # Make some changes

- » vim notes.md

- » # Status?

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   notes.md

no changes added to commit (use "git add" and/or "git commit -a")
```

GIT – GETTING STARTED (LOCAL)

- Local Repository (Repo)

- » # What changed in the file?

- » git diff

```
diff --git a/notes.md b/notes.md
index d9606bb..342449a 100644
--- a/notes.md
+++ b/notes.md
@@ -1,7 +1,7 @@

# Using git
-* git is a powerful version control system
-* git lets you use local repos
+* git is a powerful distributed version control system
+* git lets you use local and remote repos
 * git lets you version/checkpoint files to make it easy to track revisions

# Personalizing git
@@ -9,5 +9,8 @@
...
git config --global user.name "your_username"
git config --global user.email "your_email@domain.com"
+
+# Did I do this?
+git config -l
```

- » "-" = Previous version of file

- » "+" = New version of file

GIT – GETTING STARTED (LOCAL)

- Local Repository (Repo)

- » # Commit new changes
- » git add notes.md
- » git commit -m "Updated notes - better commenting"
[master 779b337] Updated notes - better commenting
1 file changed, 5 insertions(+), 2 deletions(-)
- » # History of changes:
- » git log

```
commit 779b337c302a5deb3d7fbbc71ec137c40ac76054
Author: James R. Small <james.r.small@outlook.com>
Date: Mon Sep 12 23:01:29 2016 -0400

    Updated notes - better commenting

commit 7b230679473167db5fff2f0f7c1dc32570920c81
Author: James R. Small <james.r.small@outlook.com>
Date: Mon Sep 12 22:50:49 2016 -0400

    Initial commit of my notes
```

GIT – GETTING STARTED (LOCAL)

- Local Repository (Repo)

- » # Revert to previous version - # is SHA hash from git log
- » git reset --hard 7b23067

HEAD is now at 7b23067 Initial commit of my notes

- » # Back to original version
- » more notes.md

```
# using git
* git is a powerful version control system
* git lets you use local repos
* git lets you version/checkpoint files to make it easy to track revisions

# Personalizing git
* Don't forget to configure your username and E-mail

git config --global user.name "your_username"
git config --global user.email "your_email@domain.com"
```

GIT – GETTING STARTED (LOCAL)

- Local Repository (Repo)

- » # Wait - I changed my mind again and want to switch back!

- » git reflog

- 7b23067 HEAD@{0}: reset: moving to 7b23067

- 779b337 HEAD@{1}: commit: Updated notes - better commenting

- 7b23067 HEAD@{2}: commit (initial): Initial commit of my notes

- » git reset --hard 779b337

- HEAD is now at 779b337 Updated notes - better commenting

- » more notes.md

```
# using git
* git is a powerful distributed version control system
* git lets you use local and remote repos
* git lets you version/checkpoint files to make it easy to track revisions

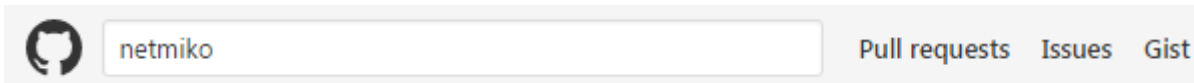
# Personalizing git
* Don't forget to configure your username and E-mail
...
git config --global user.name "your_username"
git config --global user.email "your_email@domain.com"

# Did I do this?
git config -l
```

GIT – GETTING STARTED (REMOTE)

- Start with remote repo

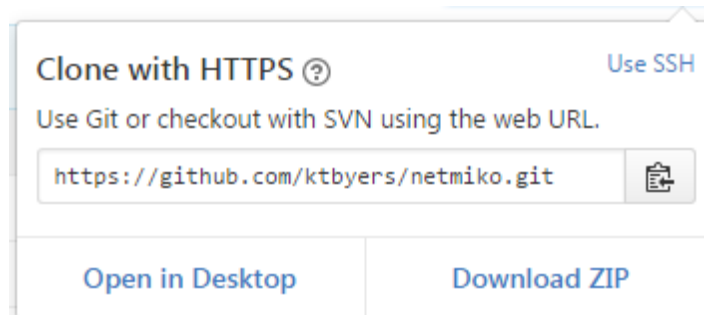
- » Search GitHub for the project repo you want:



- » In the project repo, click on the Clone or download button:



- » Copy the URL:



GIT – GETTING STARTED (REMOTE)

- Start with remote repo

- » Paste the copied URL into the terminal after git clone:

- » `git clone https://github.com/ktbyers/netmiko.git`

Cloning into 'netmiko'...

remote: Counting objects: 4244, done.

remote: Compressing objects: 100% (11/11), done.

remote: Total 4244 (delta 2), reused 0 (delta 0), pack-reused 4233

Receiving objects: 100% (4244/4244), 756.93 KiB | 0 bytes/s, done.

Resolving deltas: 100% (2759/2759), done.

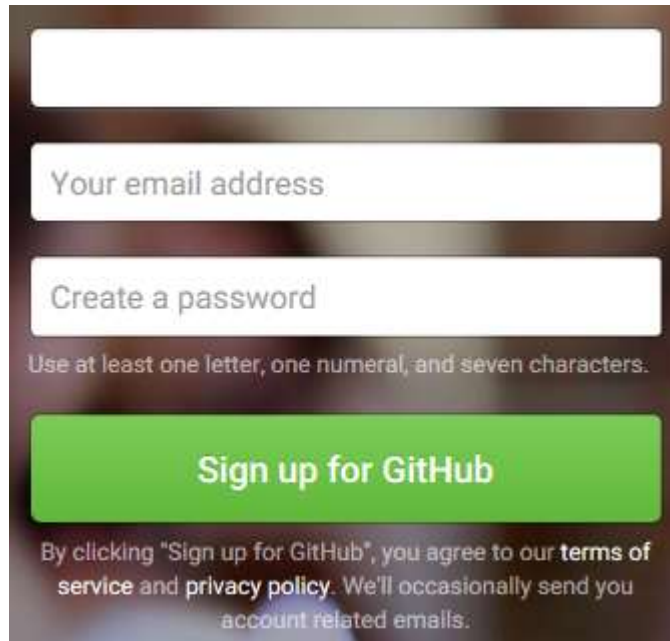
Checking connectivity... done.

- » # You now have a complete local copy of the project

- » # to play with...

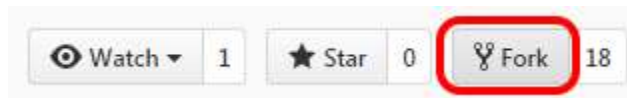
GITHUB – GETTING STARTED

- Open your favorite browser and navigate to `github.com` – sign up for an account, it's free:

A screenshot of the GitHub sign-up form. It features three input fields: a top field for a username, a middle field labeled "Your email address", and a bottom field labeled "Create a password". Below the password field is a small text requirement: "Use at least one letter, one numeral, and seven characters." A prominent green button labeled "Sign up for GitHub" is positioned below the fields. At the bottom, a line of text states: "By clicking 'Sign up for GitHub', you agree to our **terms of service** and **privacy policy**. We'll occasionally send you account related emails."

GITHUB – GETTING STARTED

- Once you're logged in to GitHub, navigate to:
`github.com/sockduct/intro_to_sprinting`
- In the upper right hand corner, you'll see a "Fork" Icon – click on it to make a copy of the repo in your GitHub account:



Note: This is based on Chalmer Lowe's Intro_to_Sprinting Project

GITHUB – GETTING STARTED

- Once you fork a copy of this repo, you'll see a link under it referring back to the original. In my case, it points to Chalmer Lowe's repo. In your case, it will point to my repo: (why?)



- After forking the repo, from your system do a git clone:
» https://github.com/<your_github_id>/intro_to_sprinting.git

GIT/GITHUB – COLLABORATION WORKFLOW

- Cloning the repo on your system allows you to make changes:
 - » # Edit the authors.md file
 - » `vim authors.md`
 - » # Add your name to the list of authors
 - » # Checking with `git status`, you'll see the
 - » # `authors.md` file needs to be checked in:
 - » `git add authors.md`
 - » `git commit -m "Added my name to authors file"`

GIT/GITHUB – COLLABORATION WORKFLOW

- Now that you've made a change to the authors.md file and committed it, you need to push these changes back to your GitHub repo:
 - » `git push origin master`
- When you clone a repo, it creates an alias called origin that points to where you cloned the repo from:
 - » `git remote -v`

GIT/GITHUB – COLLABORATION WORKFLOW

- Back on your GitHub repo page, you can now submit a Pull Request (PR). This lets you submit your changes back to the author's repo (the one you cloned from):



- » Make sure the base fork is sockduct/intro_to_sprinting and the head fork is your repo
- » Make sure you get a green check mark and "Able to merge" (what happens if there are conflicts?)

GIT/GITHUB – COLLABORATION WORKFLOW

- Click on "Create pull request:"

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub 'Comparing changes' interface. At the top, there are four dropdown menus: 'base fork: chalmerlowe/intro_to_sprinting', 'base: master', 'head fork: sockduct/intro_to_sprinting', and 'compare: master'. These are all enclosed in a red rounded rectangle. Below this, a green checkmark icon is followed by the text 'Able to merge. These branches can be automatically merged.', which is also enclosed in a red rounded rectangle. A red arrow points from this text down to a green button labeled 'Create pull request' in a rounded rectangle. Another red arrow points from the 'compare: master' dropdown menu down to the same 'Create pull request' button. The button is located in a yellow box that also contains the text 'Discuss and review the changes in this comparison with others.'

base fork: chalmerlowe/intro_to_sprinting ▾ base: master ▾ ... head fork: sockduct/intro_to_sprinting ▾ compare: master ▾

✓ Able to merge. These branches can be automatically merged.

Create pull request Discuss and review the changes in this comparison with others.

GIT/GITHUB – COLLABORATION WORKFLOW

- Next, Write a summary title describing the changes
 - » Write a brief description of what you're changing in this pull request
 - » Click on "Create pull request:"

base fork: chalmerlowe/intro_to_sprinting | base: master | ... | head fork: sockduct/intro_to_sprinting | compare: master

✓ Able to merge. These branches can be automatically merged.

Merge in images directory and updated GitHub Overview

Write | Preview

Changes:

- * Create images directory to store graphics files
- * Add images used in GitHub Overview file
- * Update GitHub Overview file

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

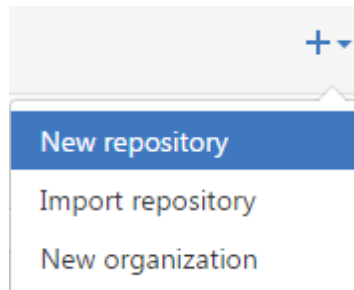
Create pull request

GIT/GITHUB – COLLABORATION WORKFLOW

- Finally, the repo owner you forked from will see the PR (Pull Request). If there are no conflicts and it makes sense, generally they will accept it which merges the changes in.
- Congratulations on participating in the Open Source process!

GITHUB – CREATING YOUR OWN REPO

- From your GitHub account, in the upper right corner click on the “+” Icon and select “New Repository”:



GITHUB – CREATING YOUR OWN REPO



- Choose a repo name, a comment and click “Create repository”:

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name


 sockduct ▾ / My-Test-Repo 

Great repository names are short and memorable. Need inspiration? How about **musical-broccoli**.

Description (optional)

An example repo for testing

☒  **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 let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾ | Add a license: **None** ▾ 

Create repository

GIT/GITHUB – OWN REPO WORKFLOW

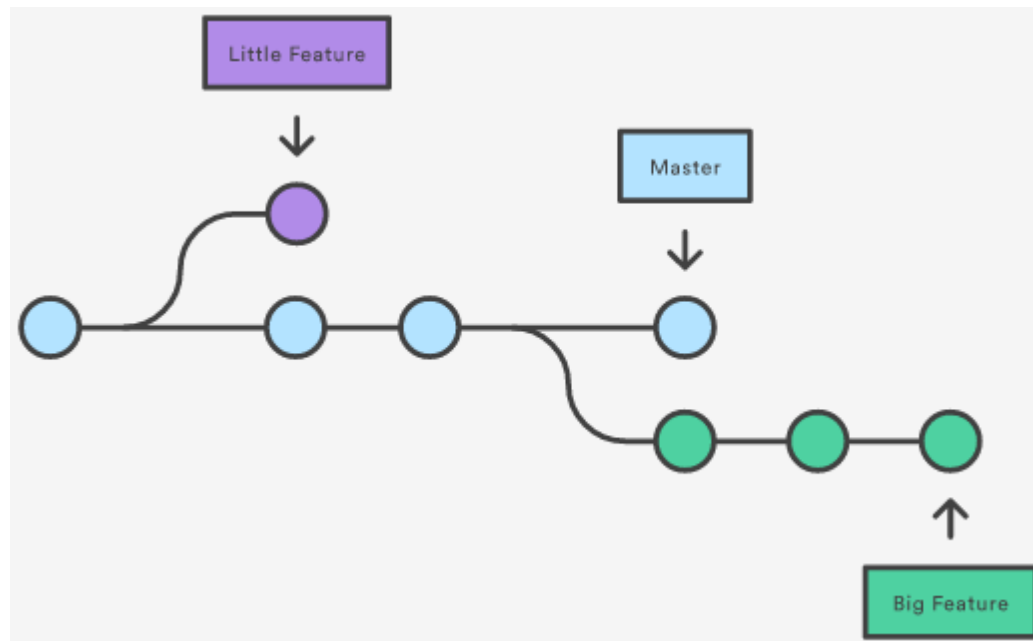
- From your system, clone the repo with git:
 - » git clone <https://github.com/<your-github-id>/My-Test-Repo.git>
- Make changes, commit, push, repeat...

GIT/GITHUB – WORKFLOW QUESTIONS

- Can't I just create the GitHub repo from git so I can use one tool?
- Why use GitHub if it's just me?
- What's a private repo? What's the difference between private and public?

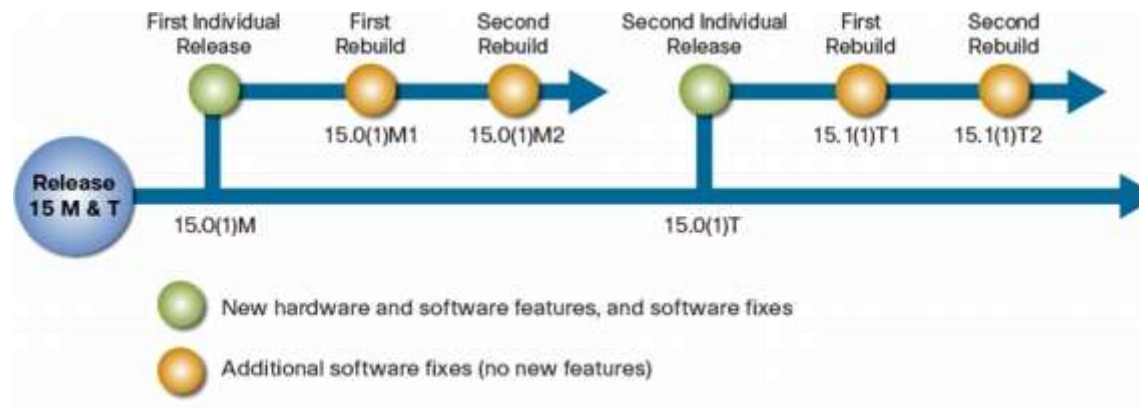
GIT/GITHUB – BRANCHES

- Branches are a way to essentially create a copy of your repo (project) for experimentation, testing or new features. The idea is to develop in the new branch without impacting the main branch:



GIT/GITHUB – BRANCHES, EXAMPLES

- For example, Cisco - a major infrastructure vendor - uses branches for its IOS software releases:



GIT/GITHUB – BRANCHES, EXAMPLE

- From your git clone of your GitHub test repo:
 - » git branch
 - * master
 - » # This is showing us there is only one branch named "master" which is the default
 - » git branch simple
 - » # Create a new branch named "simple"
 - » git branch
 - * master
 - simple

GIT/GITHUB – BRANCHES, EXAMPLE

- Use another branch:

- » # Switch branches from master to simple:

- » git checkout simple

Switched to branch 'simple'

- » # Revert to origin notes file in branch simple

- » git reflog

(...)

7b23067 HEAD@{#}: commit (initial): Initial commit of my notes

- » # Find initial commit, # will be some number, revert:

- » git reset --hard 7b23067

```
# Using git
* git is a powerful version control system
* git lets you use local repos
* git lets you version/checkpoint files to make it easy to track revisions

# Personalizing git
* Don't forget to configure your username and E-mail
...
git config --global user.name "your_username"
git config --global user.email "your_email@domain.com"
```


GIT/GITHUB – BRANCHES, EXAMPLE

- But, we haven't thrown away the updated version:
 - » # Switch branches from simple to master:
 - » `git checkout master`Switched to branch 'master'
- Check file again – it's the "updated" version:

```
# using git
* git is a powerful distributed version control system
* git lets you use local and remote repos
* git lets you version/checkpoint files to make it easy to track revisions

# Personalizing git
* Don't forget to configure your username and E-mail
...
git config --global user.name "your_username"
git config --global user.email "your_email@domain.com"

# Did I do this?
git config -l
```

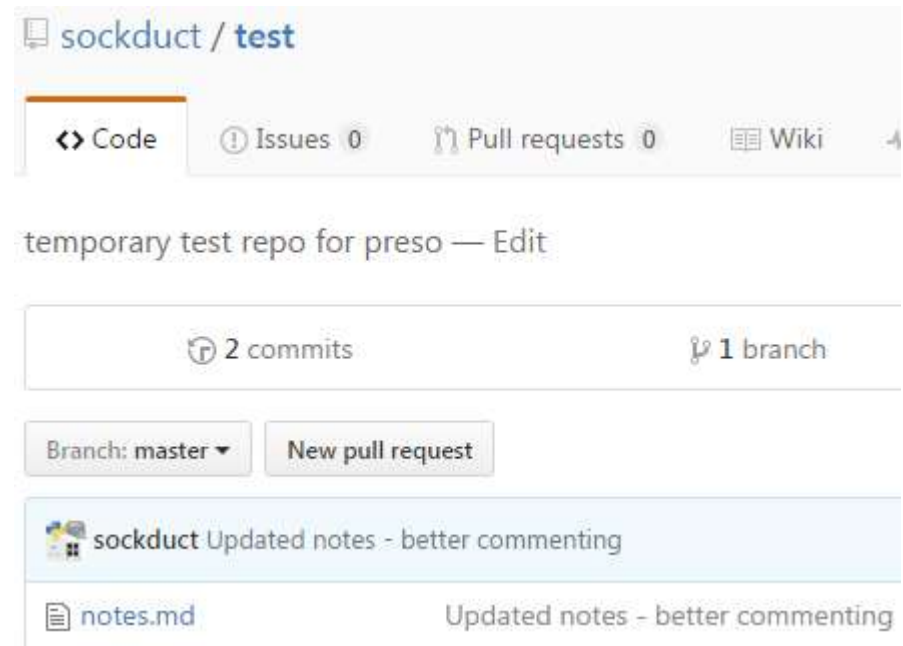
Note: To delete a branch: `git branch -d <branch>`

GIT/GITHUB – PUSH REPO TO GITHUB

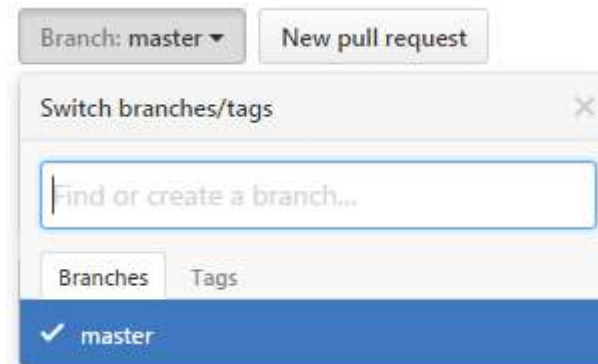
- I have a local repo with multiple branches. What if I want to push it to GitHub?
 - Recommendation: Create a repo on GitHub with the same name as your local repo directory
 - Configure local repo to use GitHub repo and push/sync it:
 - » `git remote add origin https://github.com/<your_github_id>/test.git`
 - » `git push origin master`
- Username for 'https://github.com': `<your_github_id>`
- Password for 'https://sockduct@github.com': `<your_github_password>`

GIT/GITHUB – PUSH REPO TO GITHUB

- Look on GitHub:

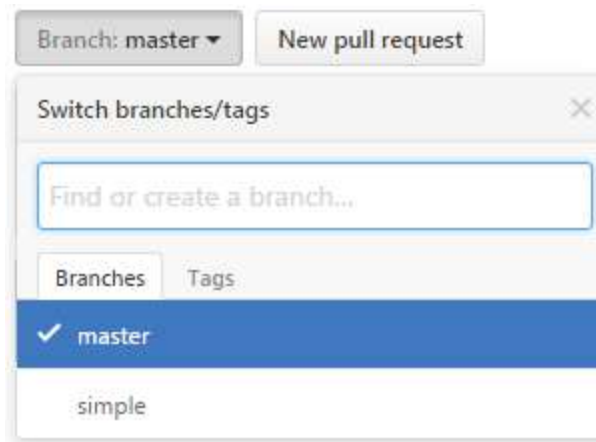


- But – where's the simple branch?



GIT/GITHUB – PUSH REPO TO GITHUB

- Remember – have to push each branch:
 - » `git push origin simple`
- Now Check GitHub repo again – may have to refresh page:



GIT/GITHUB – COLLABORATION

- What if I clone a repo on GitHub, make some changes, submit a PR and discover there's a conflict?
- You should plan to resolve the conflict yourself, it's unlikely the repo owner will do it for you:
 - » # You used git to clone your repo
 - » git remote -v
 - origin https://github.com/<your_github_id>/intro_to_sprinting (fetch)
 - origin https://github.com/<your_github_id>/intro_to_sprinting (push)
 - » # You need to pull updates from the original repo
 - » git remote add upstream https://github.com/sockduct/intro_to_sprinting

GIT/GITHUB – COLLABORATION

- Now, sync with the original (upstream) repo:
 - » # Assuming master branch:
 - » `git pull upstream master`
- Resolve any merge conflicts and push the changes back to your repo
- Now, re-submit a PR and there shouldn't be any conflicts

GIT/GITHUB – CALL TO ACTION

- Install git – Windows, OS X, Linux
- Setup a GitHub account
- Fork `intro_to_sprinting` repo from my account
(github.com/sockduct/intro_to_sprinting)
- Clone this repo to your system with git
- Add yourself to the authors file
- Push the change back to your GitHub repo
- Open a PR for me to accept the change
- Start using git/GitHub for projects!

QUESTIONS



[@sockduct](#)



github.com/sockduct



Appendix

GIT QUICK HELP

From any operating system:

- CLI
 - » `git --help`
 - » `git <command> --help`
 - » (e.g., `git config --help`)
- CLI, but launches web or man page:
 - » `git help git`
 - » `git help <command>`
 - » (e.g., `git help config`)
- CLI – main commands:
 - » `git [add | am | archive | bisect | branch | bundle | checkout | cherry-pick | citool | clean | clone | commit | describe | diff | fetch | format-patch | gc | grep | gui | init | log | merge | mv | notes | pull | push | rebase | reset | revert | rm | shortlog | show | stash | status | submodule | tag | worktree]`

FOR YOUR REFERENCE

Git Info

- [Client Home Page](#)
- [Documentation](#)
- [Pro Git Book](#)

GitHub Info

- [Understanding the GitHub Flow](#)
- [Explore GitHub - Project Showcases](#)
- [Viewing Contributions on Your Profile Page](#)
- [GitHub Guides](#)
- [GitHub Training & Guides on YouTube](#)
- [GitHub Help](#)