GitHub

Introduction to Git and GitHub

Michael A. Dolan, Ph.D

May 19, 2016

Outline

I. Introduction to source control

A. History and fundamental concepts behind source control

B. Centralized vs. distributed version control

II. Introduction to Git

A. *What is Git?* Basic Git concepts and architecture

B. Git workflows: Creating a new repo (adding, committing code)

C. HEAD

D. Git commands (checking out code)

E. Master vs branch concept

F. Creating a branch/switching between branches

G. Merging branches and resolving conflicts

III. Introduction to GitHub

A. *What is GitHub?* Basic GitHub concepts

B. GitHub in practice: Distributed version control

C. Cloning a remote repo

D. Fetching/Pushing to a remote repo

E. Collaborating using Git and GitHub

What is a ‘version control system?’

- a way to manage files and directories
- track changes over time
- recall previous versions
- ‘source control’ is a subset of a VCS.

Some history of source control...

(1972) Source Code Control System (SCCS)
- closed source, part of UNIX

(1982) Revision Control System(RCS)
- open source

(1986) Concurrent Versions System (CVS)
- open source

(2000) Apache Subversion (SVN)
- open source

...more history

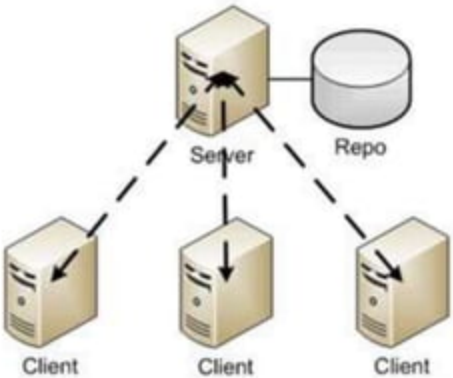
(2000) BitKeeper SCM

- closed source, proprietary, used with source code management of Linux kernel
- free until 2005
- distributed version control

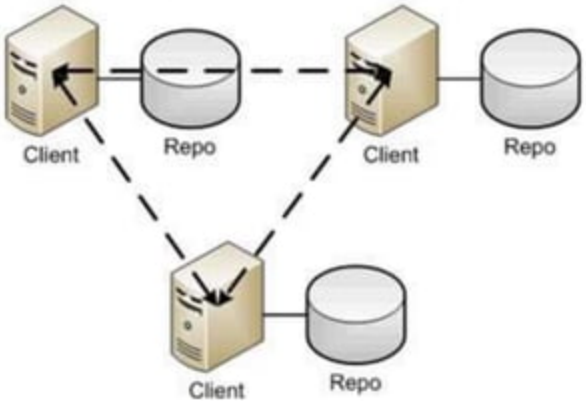
Distributed version control

No central server
Every developer is a client, the server and the repository

Traditional



Distributed



Source: http://bit.ly/1SH4E23

What is git?

What is git?

- created by Linus Torvalds, April 2005
- replacement for BitKeeper to manage Linux kernel changes
- a command line version control program
- uses checksums to ensure data integrity
- distributed version control (like BitKeeper)
- cross-platform (including Windows!)
- open source, free

Popularity

Results of the Eclipse Community Survey regarding SVN and Git usage:

Year	Git	Subversion
2009	2.4%	57.5%
2010	6.8%	58.3%
2011	12.8%	51.3%
2012	27.6%	46.0%
2013	36.3%	37.8%

https://www.openhub.net/repositories/compare

http://bit.ly/1QyLoOu

http://www.indeed.com/jobtrends/q-svn-q-git-q-subversion-q-github.html?relative=1

Git distributed version control

- "If you're not distributed, you're not worth using." – Linus Torvalds
- no need to connect to central server
- can work without internet connection
- no single failure point
- developers can work independently and merge their work later
- every copy of a Git repository can serve either as the server or as a client (and has complete history!)
- Git tracks **changes**, not versions
- Bunch of little **change sets** floating around

Is Git for me?

- People primarily working with source code
- Anyone wanting to track edits (especially changes to text files)
  - review history of changes
  - anyone wanting to share, merge changes
- Anyone not afraid of command line tools

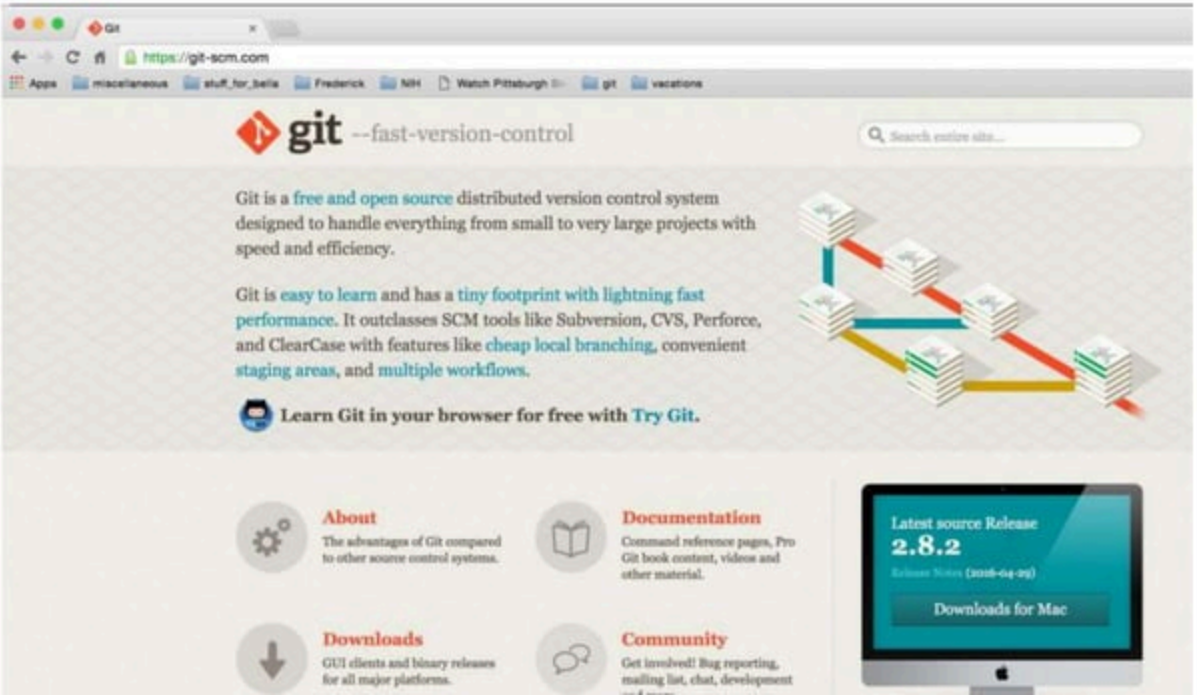
Most popular languages used with Git

- HTML
- CSS
- Javascript
- Python
- ASP
- Scala
- Shell scripts
- PHP
- Ruby
- Ruby on Rails
- Perl
- Java
- C
- C++
- C#
- Objective C
- Haskell
- CoffeeScript
- ActionScript

Not as useful for image, movies, music…and files that must be interpreted (.pdf, .psd, etc.)

How do I get it?

http://git-scm.com



Git install tip

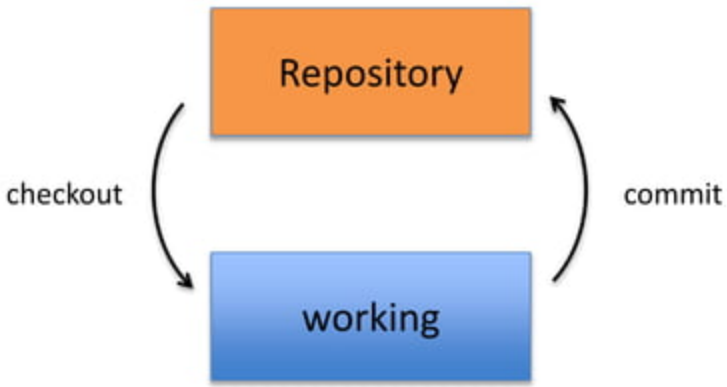
- Much better to set up on a per-user basis (instead of a global, system-wide install)

What is a repository?

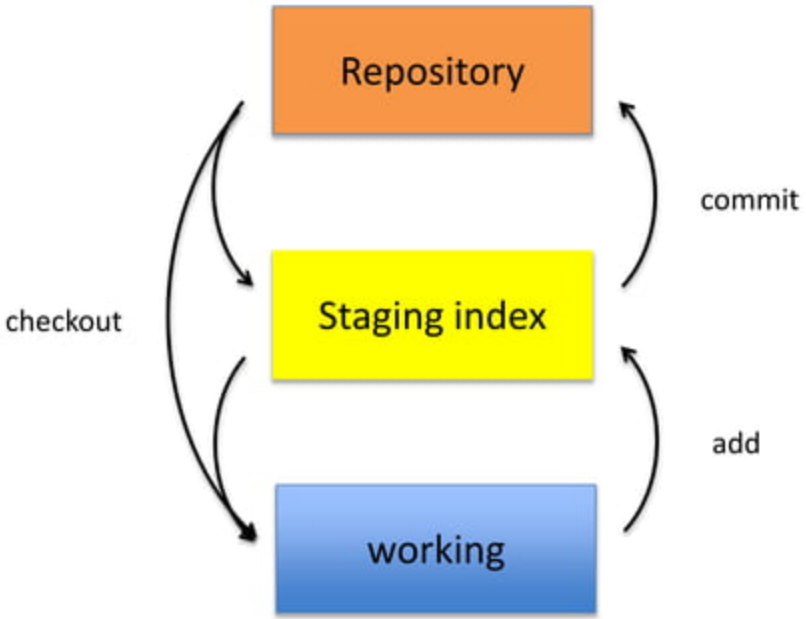
- “repo” = repository
- usually used to organize a single project
- repos can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs

Two-tree architecture

other VCSs



Git uses a three-tree architecture



A simple Git workflow

1. Initialize a new project in a directory:

   git init

```
[ dolanmi L02029756  ~/Desktop ]$ mkdir new_project
[ dolanmi L02029756  ~/Desktop ]$ cd new_project/
[ dolanmi L02029756  ~/Desktop/new_project ]$ git init
Initialized empty Git repository in /Users/dolanmi/Desktop/new_project/.git/
[ dolanmi L02029756  ~/Desktop/new_project ]$
```

2. Add a file using a text editor to the directory
3. Add every change that has been made to the directory:

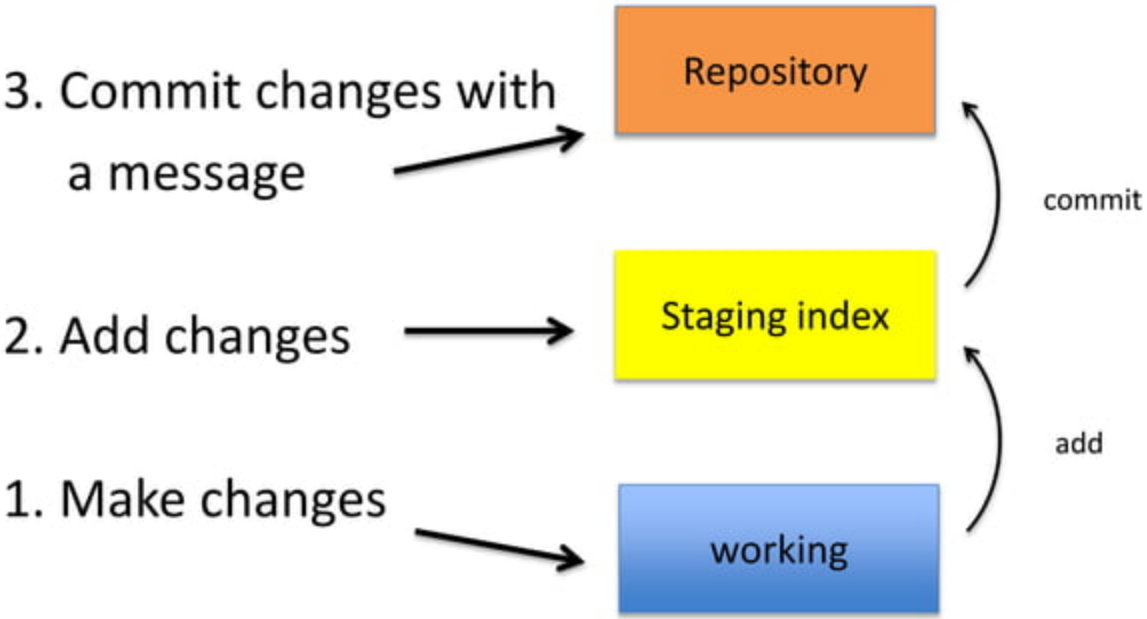
   git add .

4. Commit the change to the repo:

   git commit –m “important message here”

```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git add .
[ dolanmi L02029756  ~/Desktop/new_project ]$ git commit -m "Add message to file.txt"
[master (root-commit) 1a7e4a5] Add message to file.txt
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
[ dolanmi L02029756  ~/Desktop/new_project ]$
```

After initializing a new git repo…



A note about commit messages

- Tell what it does (present tense)
- Single line summary followed by blank space followed by more complete description
- Keep lines to <= 72 characters
- Ticket or bug number helps

Good and bad examples

Bad: "Typo fix"

Good: "Add missing / in CSS section"

Bad: "Updates the table. We'll discuss next Monday with Darrell."

Bad: `git commit -m "Fix login bug"`

Good: `git commit -m`

```
Redirect user to the requested page after login

https://trello.com/path/to/relevant/card

Users were being redirected to the home page after login, which is less
useful than redirecting to the page they had originally requested before
being redirected to the login form.

* Store requested path in a session variable
* Redirect to the stored location after successfully logging in the user
```

How to I see what was done?

git log

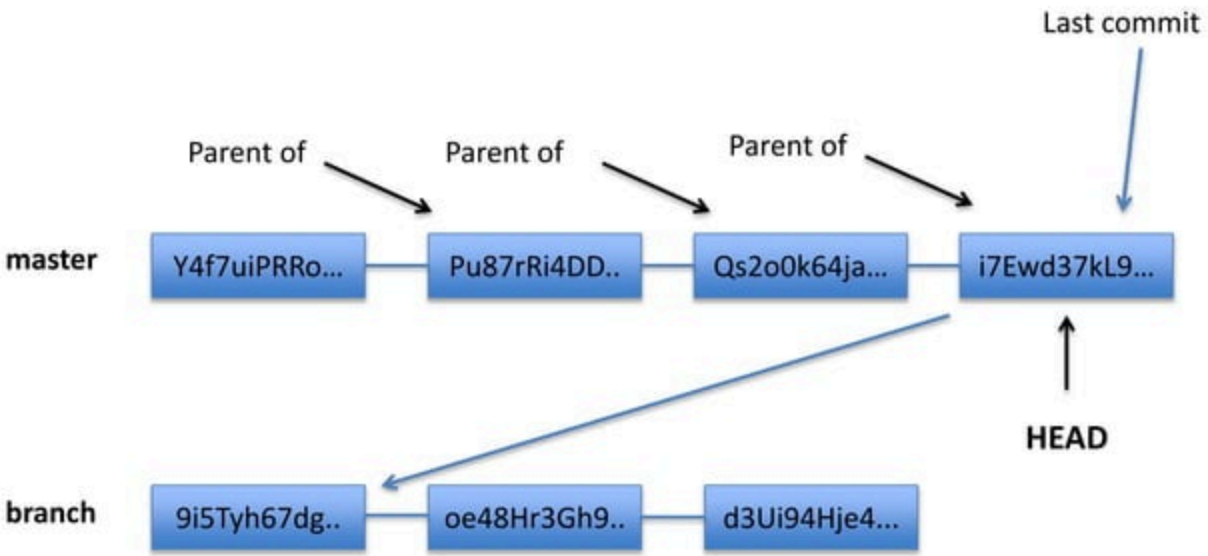
```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git log
commit 6c40ffd9ba4ba1567eb6fcd3715f12a15b0a678d
Author: mchldln <dolanmi@niaid.nih.gov>
Date:   Mon May 2 18:11:23 2016 -0400

    Add message to text file
[ dolanmi L02029756  ~/Desktop/new_project ]$
```

```
[ dolanmi L02029756  ~/Desktop/bcbb/portal_project/git/BCBBportalXI ]$ git log
commit f8c00639a649a122446040b15185cc09c4c5c71c
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Fri Apr 29 15:02:56 2016 -0400

    update headers

commit eb0cf49cc05786cbc7314982f06af5a9ad93149e
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Tue Apr 26 12:07:32 2016 -0400

    update name link and about page

commit 44c433a1794cfef211d5116568dcfbe67d518b2f
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Mon Apr 25 15:45:27 2016 -0400

    remove about, change font family in the name

commit 898be0093a995c08a7a4f99219abee255b94a874
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Fri Apr 22 09:30:49 2016 -0400

    updating header and sidenav bar

commit c5f689ed0b8c71582b3d301e2282f9e6472962c6
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Thu Apr 21 14:29:20 2016 -0400

    change the name to code

commit 4463ea2d1c75b80af9d2894feb2eb3ded7fe40c9
:
commit f8c00639a649a122446040b15185cc09c4c5c71c
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Fri Apr 29 15:02:56 2016 -0400

    update headers

commit eb0cf49cc05786cbc7314982f06af5a9ad93149e
Author: Yamil Boo <yamil.booirizarry@nih.gov>
Date:   Tue Apr 26 12:07:32 2016 -0400

    update name link and about page
```

Checksums generated by SHA1 encryption algorithm

The HEAD pointer

- points to a specific commit in repo
- as new commits are made, the pointer changes
- **HEAD always points to the "tip" of the currently checked-out branch in the repo**
- (not the working directory or staging index)
- last state of repo (what was checked out initially)
- HEAD points to parent of next commit (where writing the next commit takes place)


Last commit
Parent of
Parent of
Parent of
master
Y4f7uiPRRo...
Pu87rRi4DD..
Qs2o0k64ja...
i7Ewd37kL9...
HEAD
branch
9i5Tyh67dg..
oe48Hr3Gh9..
d3Ui94Hje4...

Which files were changed and where do they sit in the three tree?

git status – allows one to see where files are in the three tree scheme

```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git 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:   file.txt

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

```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file.txt
```

What changed in working directory?

git diff – compares changes to files between repo and working directory

```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git diff
diff --git a/file.txt b/file.txt
index 4e1c952..bd5fd23 100644
--- a/file.txt
+++ b/file.txt
@@ -1 +1 @@
-NIEHS is not great!
+NIEHS is great!
```

Line numbers in file

Removed

Added

Note: git diff --staged - compares staging index to repo

Note: git diff filename can be used as well

Deleting files from the repo

git rm filename.txt

- moves deleted file change to staging area

- It is not enough to delete the file in your working directory. You must commit the change.

Deleting files from the repo

```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git add file.txt
[ dolanmi L02029756  ~/Desktop/new_project ]$ git commit -m "message"
[master (root-commit) 1edeae8] message
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
nothing to commit, working directory clean
[ dolanmi L02029756  ~/Desktop/new_project ]$ git rm file.txt
rm 'file.txt'
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    file.txt

[ dolanmi L02029756  ~/Desktop/new_project ]$ git commit -m "delete file.txt"
[master c4f8073] delete file.txt
 1 file changed, 1 deletion(-)
 delete mode 100644 file.txt
```

```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
nothing to commit, working directory clean
```

Moving (renaming) files

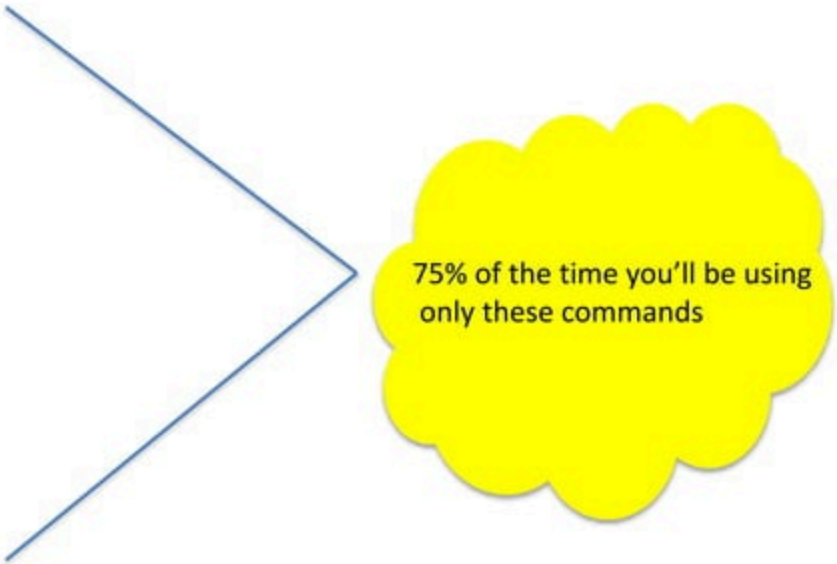
git mv filename1.txt filename2.txt

```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git mv file2.txt file1.txt
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        renamed:    file2.txt -> file1.txt
```

Note: File file1.txt was committed to repo earlier.

Good news!

git init
git status
git log
git add
git commit
git diff
git rm
git mv



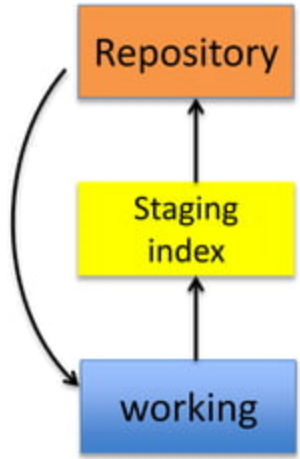
What if I want to undo changes made to working directory?

git checkout *something*

(where "something" is a file or an entire branch)

- git checkout will grab the file from the repo
- *Example:* git checkout -- file1.txt

("checkout file 'file1.txt' from the current branch")



What if I want to undo changes added to staging area?

git reset HEAD filename.txt

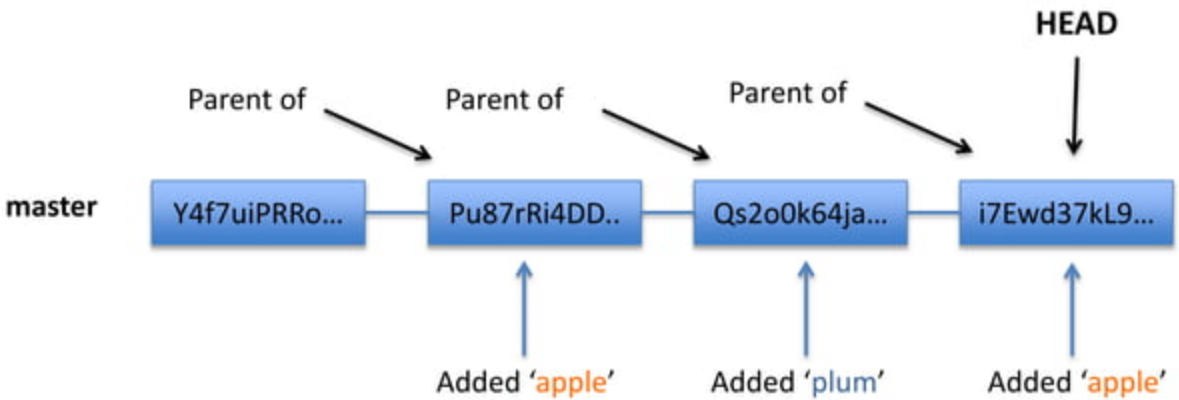
```
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ vi file4.txt
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git add .
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file4.txt

[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git reset HEAD file4.txt
Unstaged changes after reset:
M       file4.txt
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git 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:   file4.txt

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

What if I want to undo changes committed to the repo?

git commit --amend -m “message”

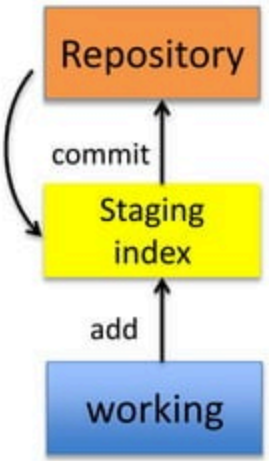
- allows one to amend a change to the last commit
- anything in staging area will be amended to the last commit

Note: To undo changes to older commits, make a new commit



Obtain older versions

```
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git log
commit 60f1c1a034fdcab4e8127d36556a881e7778c2ec
Author: mchldln <dolanmi@niaid.nih.gov>
Date:   Tue May 3 17:00:36 2016 -0400

    another message yet

commit d685ff9a41a9eec62e6010827513e33ba1abc0d6
Author: mchldln <dolanmi@niaid.nih.gov>
Date:   Tue May 3 17:00:09 2016 -0400

    another message

commit 6e073c640928b1470f8443e594fb63063c87bcf7
Author: mchldln <dolanmi@niaid.nih.gov>
Date:   Tue May 3 14:25:38 2016 -0400

    message
```



git checkout 6e073c640928b -- filename.txt

Note: Checking out older commits places them into the staging area

git checkout 6e073c640928b -- filename.txt

```
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git checkout 6e073c640928b -- file4.txt
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file4.txt

[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git diff
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git diff --staged
diff --git a/file4.txt b/file4.txt
index 56392a0..9c595a6 100644
--- a/file4.txt
+++ b/file4.txt
@@ -1 +1 @@
-temp temp temp 2
+temp
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git reset HEAD file4.txt
Unstaged changes after reset:
M       file4.txt
```

Which files are in a repo?

git ls-tree tree-ish

tree-ish – a way to reference a repo

full SHA, part SHA, HEAD, others

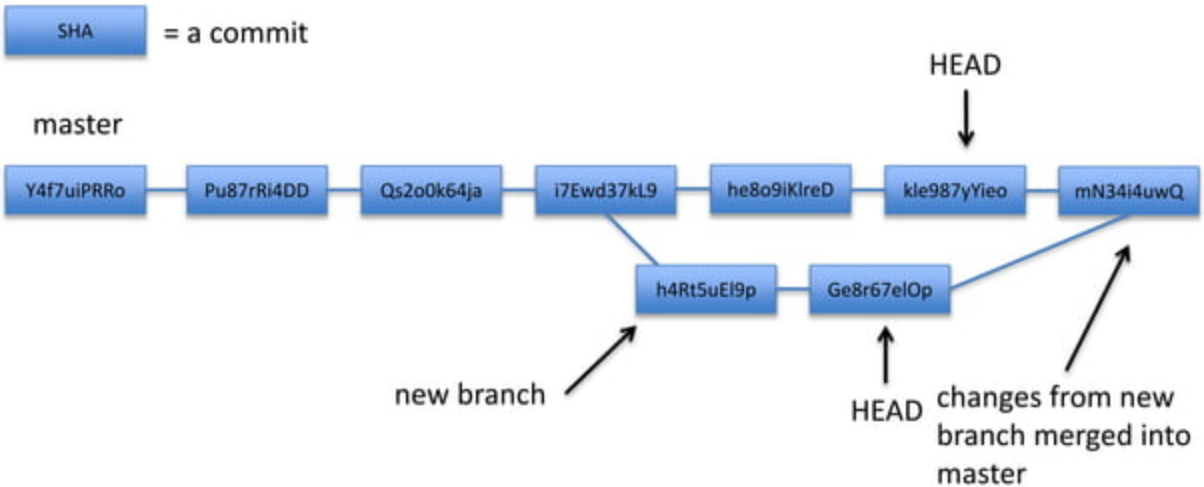
```
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git ls-tree HEAD
100644 blob 5626abf0f72e58d7a153368ba57db4c673c0e171    file1.txt
100644 blob f719efd430d52bcfc8566a43b2eb655688d38871    file2.txt
100644 blob a5648e79c58aab29ec5e45e99781edd7263e19e7    file3.txt
100644 blob 9c595a6fb7692405a5c4a10e1caf93d7a5bd9c37    file4.txt
040000 tree 6460ee80311f76a04b884e60f25400cf30b477b9    sub_dir
```

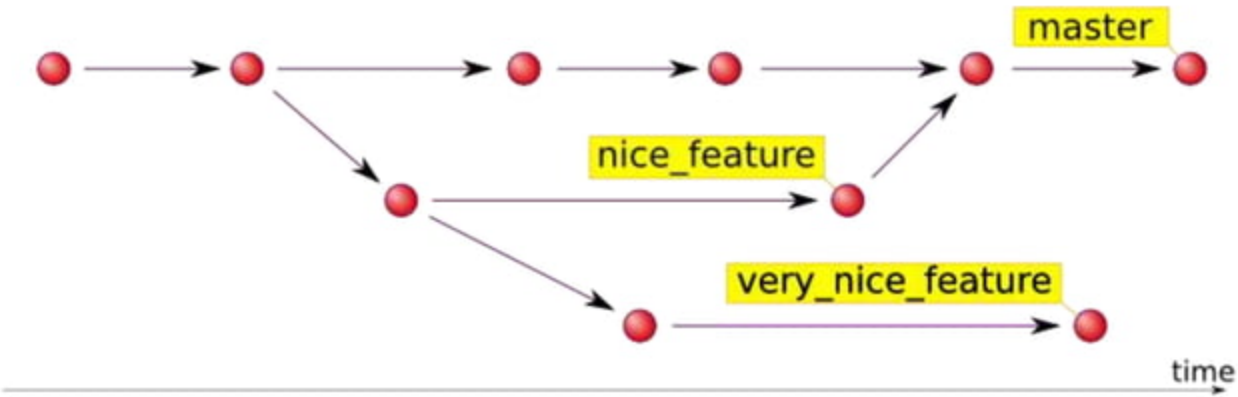
blob = file, tree = directory

branching

- allows one to try new ideas
- If an idea doesn't work, throw away the branch. Don't have to undo many changes to master branch
- If it *does* work, merge ideas into master branch.
- There is only one working directory

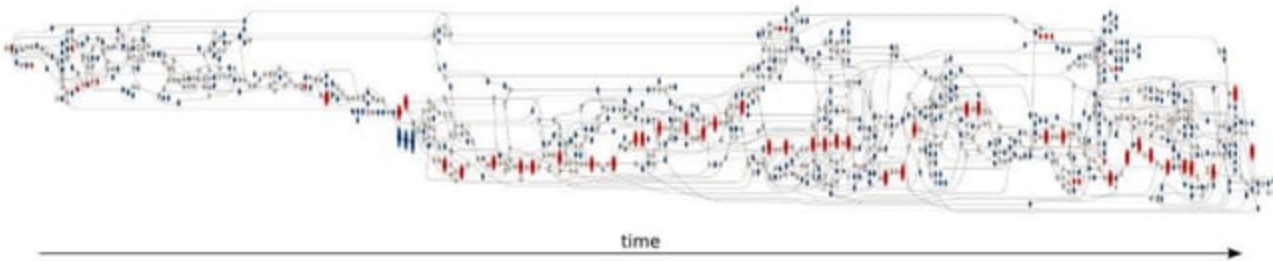
Branching and merging example





Source: http://hades.github.io/2010/01/git-your-friend-not-foe-vol-2-branches/

16 forks and 7 contributors to the master branch



red - commits pointed to by tags
blue - branch heads
white - merge and bifurcation commits.

Source: https://www.tablix.org/~avian/blog/archives/2014/06/vesna_drivers_git_visualization/

In which branch am I?

git branch

```
[dolanmi]$ git branch
* master
```

How do I create a new branch?

git branch new_branch_name

```
[dolanmi]$ git branch
* master
  new_feature
```

Note: At this point, both HEADs of the branches are pointing to the same commit (that of master)

How do I switch to new branch?

git checkout new_branch_name

```
[dolanmi]$ git checkout new_feature
Switched to branch 'new_feature'
[dolanmi]$ git branch
  master
* new_feature
```

At this point, one can switch between branches, making commits, etc. in either branch, while the two stay separate from one another.

Note: In order to switch to another branch, your current working directory must be clean (no conflicts, resulting in data loss).

Comparing branches

git diff first_branch..second_branch

```
[dolanmi]$ git diff master..new_feature
diff --git a/file1.txt b/file1.txt
index 5626abf..1684a0f 100644
--- a/file1.txt
+++ b/file1.txt
@@ -1 +1 @@
-one
+new information
```

How do I merge a branch?

From the branch into which you want to merge another branch....

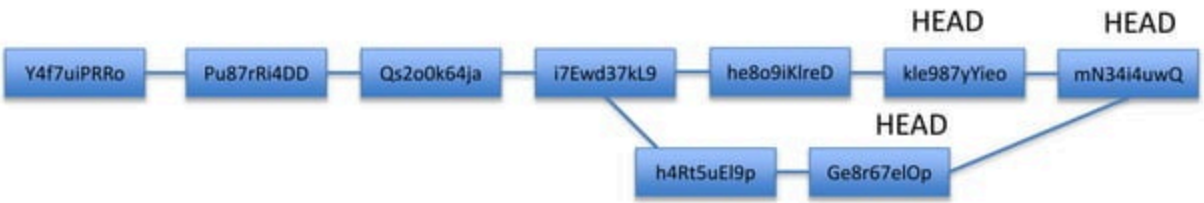
git merge branch_to_merge

```
[dolanmi]$ git branch
* master
  new_feature
[dolanmi]$ git merge new_feature
Updating 3789cd3..1214807
Fast-forward
 file1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[dolanmi]$ git diff master..new_feature
[dolanmi]$
```

Note: Always have a clean working directory when merging

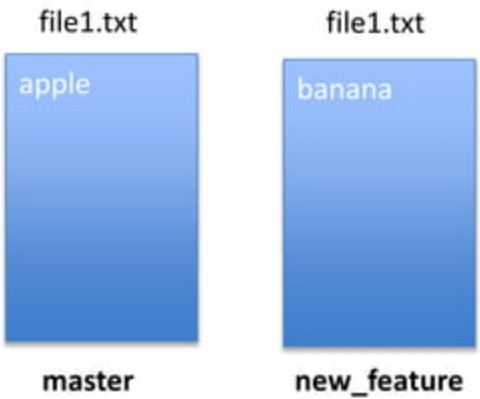
"**fast-forward**" merge occurs when HEAD of master branch is seen when looking back

"**recursive**" merge occurs by looking back and combining ancestors to resolve merge



merge conflicts

What if there are two changes to same line in two different commits?




```
[dolanmi]$ git merge new_feature
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Resolving merge conflicts

Git will notate the conflict in the files!

```
<<<<<<< HEAD
apple
=======
banana
>>>>>>> new_feature
```

Solutions:

1. Abort the merge using git merge –abort
2. Manually fix the conflict
3. Use a merge tool (there are many out there)

Graphing merge history

git log --graph --oneline --all --decorate

```
[dolanmi]$ git log --graph --oneline --all --decorate
*   7367e1e (HEAD -> master) fix merge conflict
|\
| * b4f09a5 (new_feature) add banana
* | df043c1 add apple
|/
* 1214807 new information added
* 3789cd3 file3.txt
* 6bfebcd new dir
* 730c6bd files
* 48f1ecf c
* 60f1c1a another message yet
* d685ff9 another message
* 6e073c6 message
```

Tips to reduce merge pain

- merge often
- keep commits small/focused
- bring changes occurring to master into your branch frequently (“tracking”)

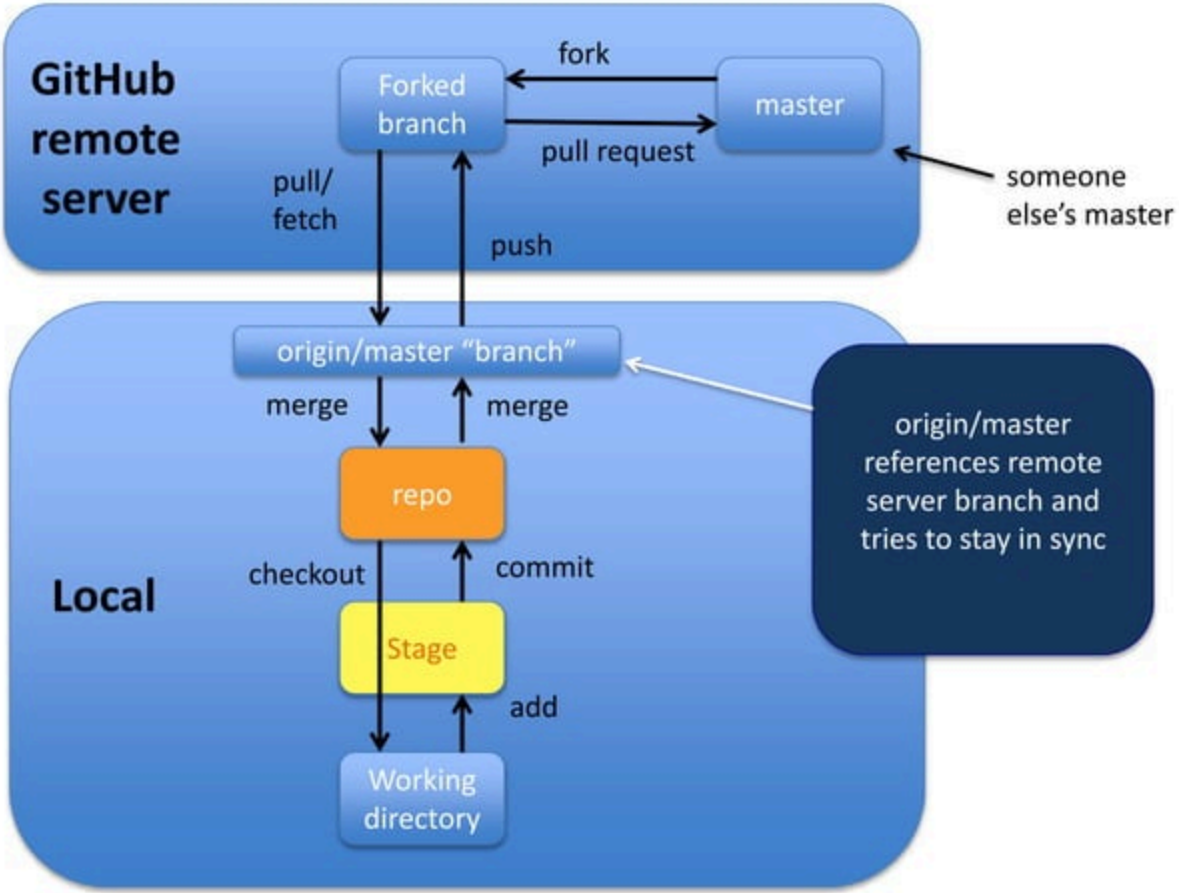
What is  ?

GitHub


Personal
Open source
Business
Explore
Pricing
Blog
Support
Sign in
Sign up
How people build software
Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.
Pick a username
Your email address
Create a password
Sign up for GitHub

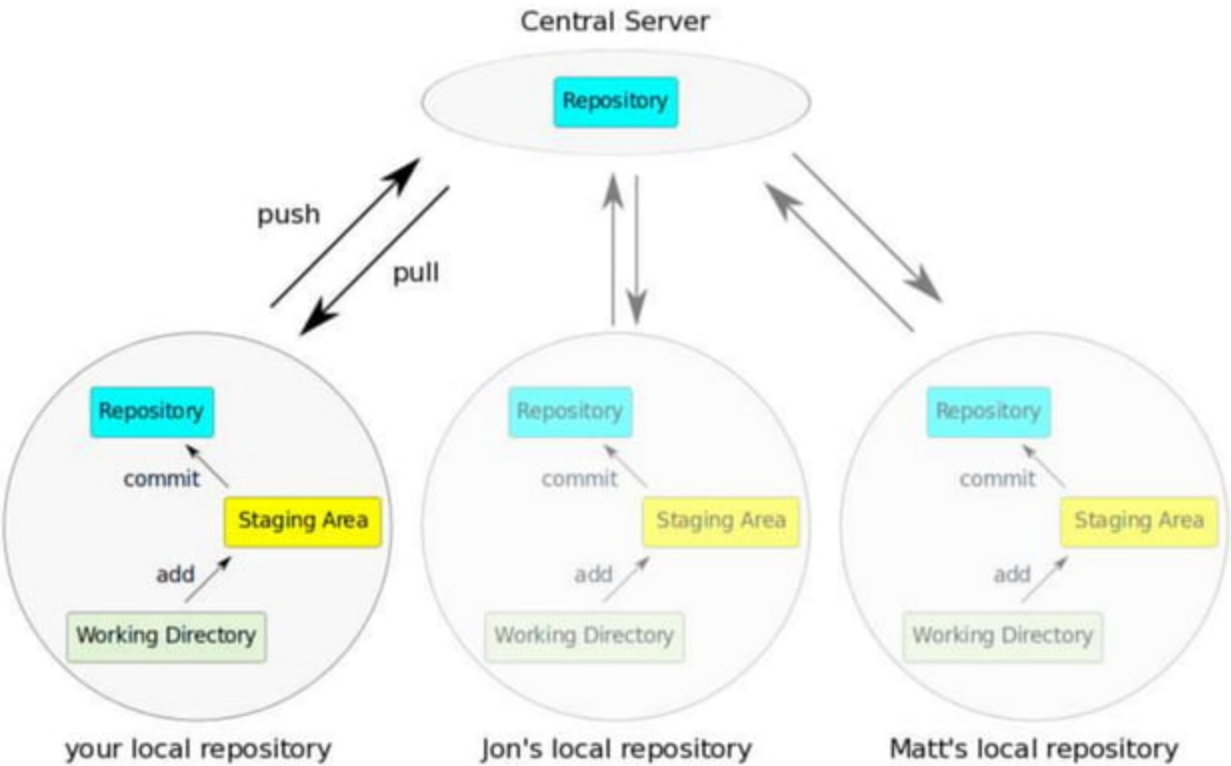
GitHub

- **a platform to host git code repositories**
- http://github.com
- launched in 2008
- most popular Git host
- allows users to collaborate on projects from anywhere
- GitHub makes git social!
- Free to start


GitHub remote server
Forked branch
fork
master
pull request
someone else's master
pull/ fetch
push
origin/master "branch"
merge
merge
repo
checkout
commit
Local
Stage
add
Working directory
origin/master references remote server branch and tries to stay in sync

Important to remember

Sometimes developers *choose* to place repo on GitHub as a centralized place where everyone commits changes, but it **doesn't have to be on GitHub**



Source: http://bit.ly/1rvzjp9

This repository Search

Pull requests Issues Gist

mchldln / open-sourcerer
forked from diy/open-sourcerer

Unwatch 1 Star 0 Fork 113

Code Pull requests 0 Wiki Pulse Graphs Settings

DIY Skill — Edit

124 commits 1 branch 0 releases 36 contributors

Branch: master New pull request New file Upload files Find file HTTPS https://github.com/mchldl Download ZIP

This branch is 2 commits ahead of diy:master. Pull request Compare

mchldln Corrected typo Latest commit ecd0d3b 6 days ago

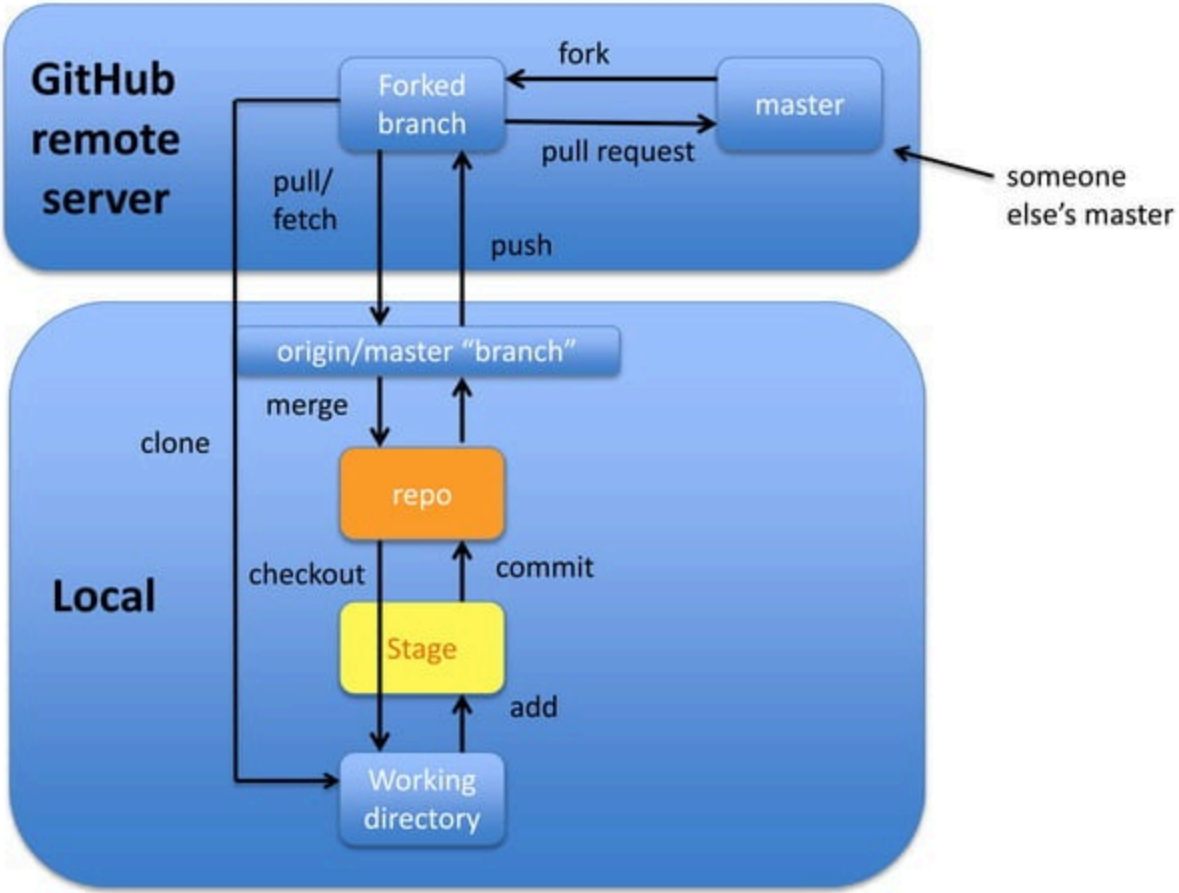
.gitignore	Add gitignore	3 years ago
README.md	added my name on README.md	2 years ago
collaborative-story.txt	Corrected typo	6 days ago
new-features.txt	Add lines to collaborative-story, new-features and ultimate-cookie vi...	2 years ago
script.md	Updated script.md	3 years ago
ultimate-cookie.txt	Add lines to collaborative-story, new-features and ultimate-cookie vi...	2 years ago

README.md

Copying (cloning) files from remote repo to local machine

git clone URL <new_dir_name>

```
[dolanmi]$ git clone  https://github.com/mchldln/open-sourcerer.git program_one
Cloning into 'program_one'...
remote: Counting objects: 294, done.
remote: Total 294 (delta 0), reused 0 (delta 0), pack-reused 294
Receiving objects: 100% (294/294), 45.83 KiB | 0 bytes/s, done.
Resolving deltas: 100% (149/149), done.
Checking connectivity... done.
[dolanmi]$ ls
program_one
[dolanmi]$ cd program_one/
[dolanmi]$ ls -aFlt
total 72
drwxrwxr-x   9 dolanmi  NIH\Domain Users    306 May  4 17:26 ./
drwxrwxr-x  13 dolanmi  NIH\Domain Users    442 May  4 17:26 .git/
-rw-rw-r--   1 dolanmi  NIH\Domain Users     19 May  4 17:26 .gitignore
-rw-rw-r--   1 dolanmi  NIH\Domain Users    586 May  4 17:26 README.md
-rw-rw-r--   1 dolanmi  NIH\Domain Users   2938 May  4 17:26 collaborative-story.txt
-rw-rw-r--   1 dolanmi  NIH\Domain Users    138 May  4 17:26 new-features.txt
-rw-rw-r--   1 dolanmi  NIH\Domain Users  12984 May  4 17:26 script.md
-rw-rw-r--   1 dolanmi  NIH\Domain Users    192 May  4 17:26 ultimate-cookie.txt
drwxrwxr-x   3 dolanmi  NIH\Domain Users    102 May  4 17:26 ../
```


GitHub remote server
Forked branch
fork
master
pull request
someone else's master
pull/ fetch
push
origin/master "branch"
merge
clone
repo
checkout
commit
Local
Stage
add
Working directory

How do I link my local repo to a remote repo?

git remote add <alias> <URL>

Note: This just establishes a connection…no files are copied/moved

Note: Yes! You may have more than one remote linked to your local directory!

Which remotes am I linked to?

git remote

Pushing to a remote repo

git push local_branch_alias branch_name

```
[dolanmi]$ git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 280 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To https://github.com/mchldln/open-sourcerer.git
   ecd0d3b..212432e  master -> master
```

Fetching from a remote repo

git fetch remote_repo_name

Fetch in no way changes a your working dir or any commits that you've made.

- Fetch before you work
- Fetch before you push
- Fetch often

git merge must be done to merge fetched changes into local branch

Collaborating with Git

This repository Search

Pull requests Issues Gist

mchldln / niehs

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

This is a test account for NIEHS. — Edit

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request New file Upload files Find file HTTPS https://github.com/mchldl Download ZIP

mchldln Update README.md Latest commit 6d509e4 on Mar 28

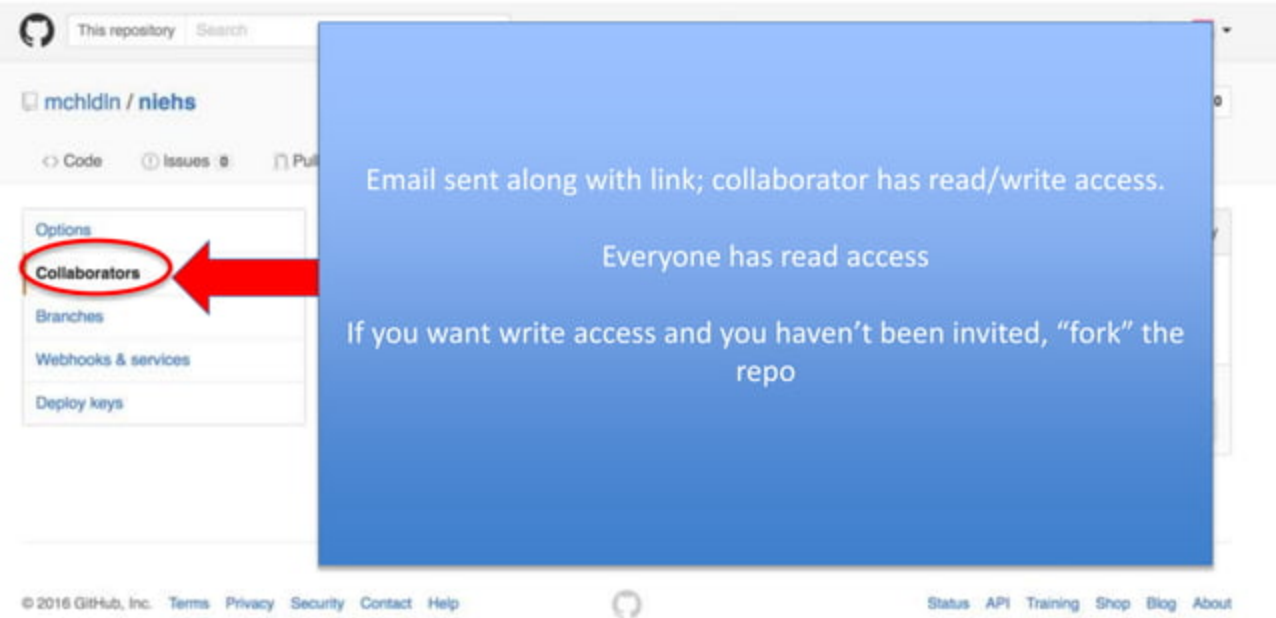
README.md Update README.md a month ago

README.md

niehs

This is a test account for NIEHS. new text

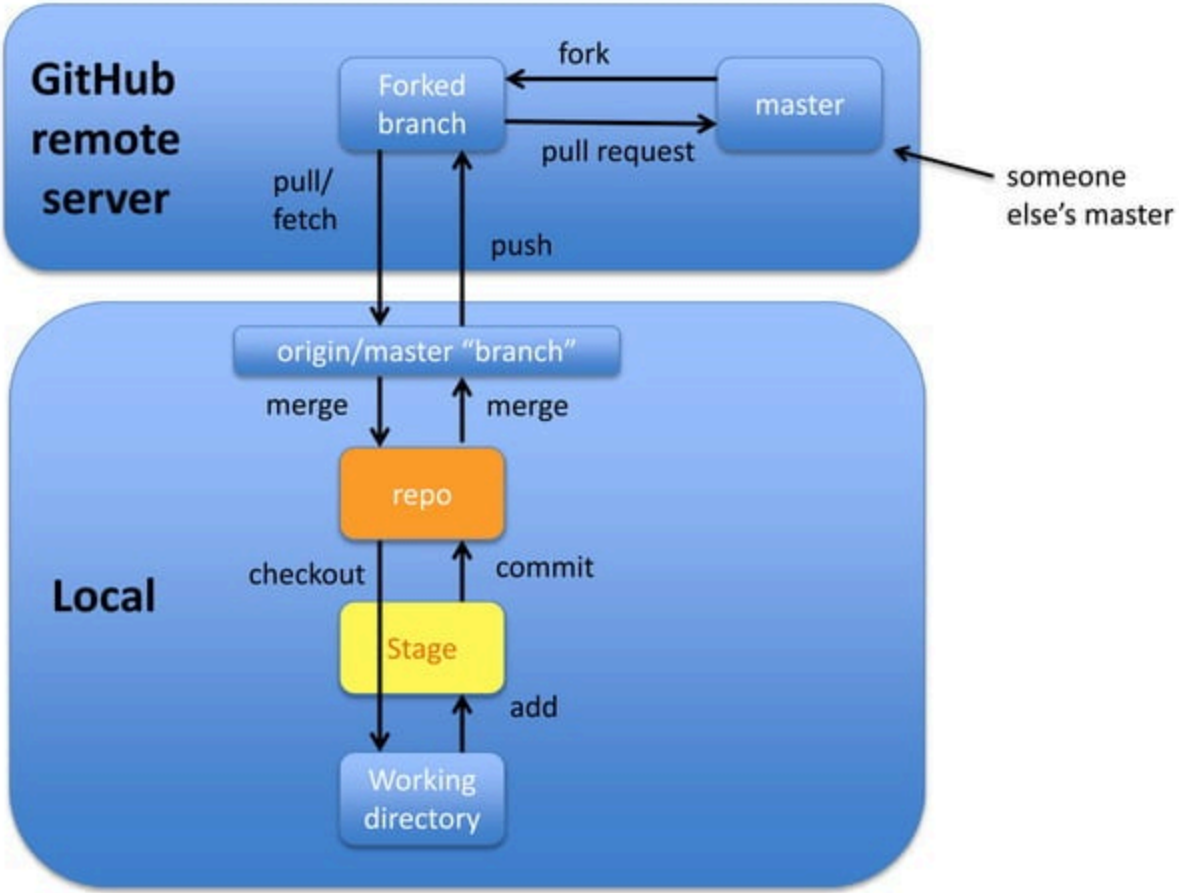
Collaborating with Git



Email sent along with link; collaborator has read/write access.

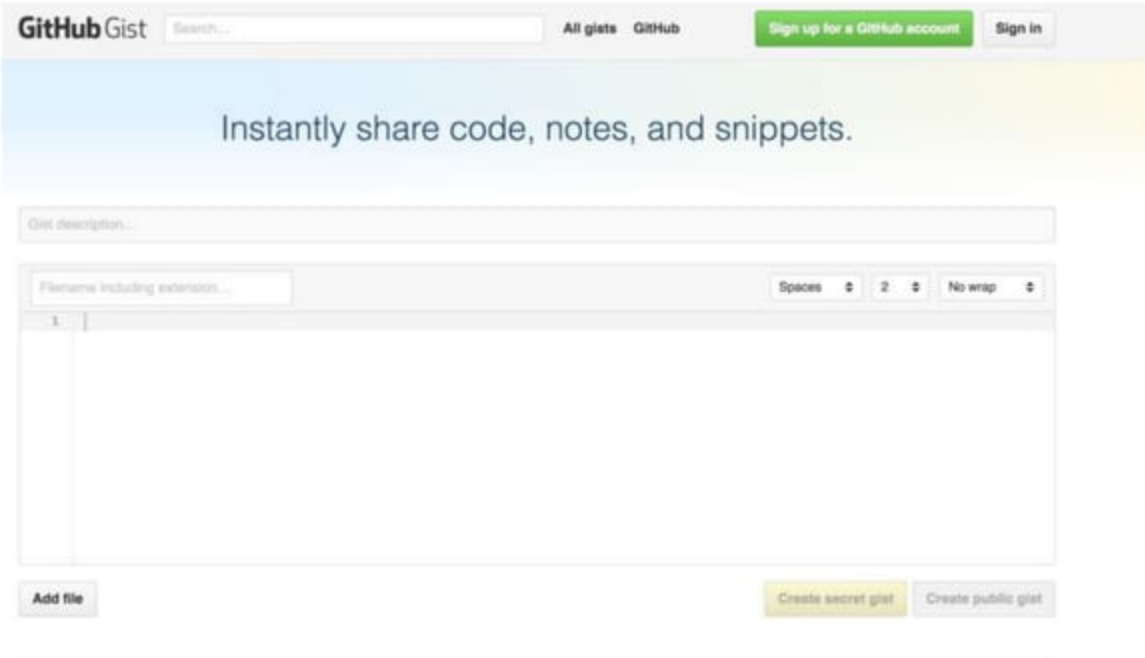
Everyone has read access

If you want write access and you haven't been invited, "fork" the repo


GitHub remote server
Forked branch
fork
master
pull request
someone else's master
pull/ fetch
push
origin/master "branch"
merge
merge
repo
checkout
commit
Local
Stage
add
Working directory

GitHub Gist

https://gist.github.com/



Good resources

- Git from Git: https://git-scm.com/book/en/v2
- A guided tour that walks through the fundamentals of Git: https://githowto.com
- Linus Torvalds on Git: https://www.youtube.com/watch?v=idLyobOhtO4
- Git tutorial from Atlassian: https://www.atlassian.com/git/tutorials/
- A number of easy-to-understand guides by the GitHub folks https://guides.github.com

git commit -a

- Allows one to add to staging index and commit *at the same time*
- Grabs everything in working direcotry
- Files not tracked or being deleted are not included

git log --oneline

- gets first line and checksum of all commits in current branch

```
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git log --oneline
3789cd3 file3.txt
6bfebcd new dir
730c6bd files
48f1ecf c
60f1c1a another message yet
d685ff9 another message
6e073c6 message
```

git diff g5iU0oPe7x

When using checksum of older commit, will show you all changes compared to those in your working directory

Renaming and deleting branches

git branch –m/--move old_name new_name

git branch –d branch_name

Note: Must not be in branch_name

Note: Must not have commits in branch_name unmerged in branch from which you are deleting

git branch –D branch_name

Note: If you are *really* sure that you want to delete branch with commits

Tagging

- Git has the ability to tag specific points in history as being important, such as releases versions (v.1.0, 2.0, …)

git tag

```
$ git tag
v0.1
v1.3
```

Tagging

Two types of tags:

lightweight – a pointer to a specific comment – basically a SHA stored in a file

```
git tag tag_name
```

annotated – a full object stored in the Git database – SHA, tagger name, email, date, message and can be signed and verified with GNU Privacy Guard (GPG)

```
git tag –a tag_name –m "message"
```

How do I see tags?

git show tag_name

```
$ git show v1.4-lw
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number
```

Lightweight tag

```
$ git show v1.4
tag v1.4
Tagger: Ben Straub <ben@straub.cc>
Date:   Sat May 3 20:19:12 2014 -0700

my version 1.4

commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number
```

Annotated tag