APLICAÇÕES INFORMÁTICAS EM EMGENHARIA BIOMÉDICA

Introduction to
Git and GitHub

Outline

I. Introduction to source control
- A. 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’ (VCS)?

A way to manage files and directories

Track changes over time

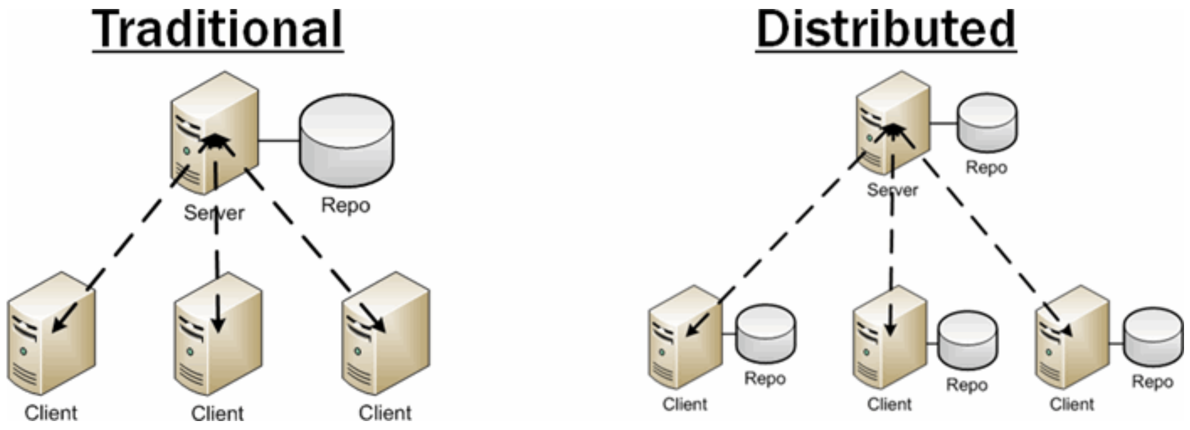
Recall previous versions

'source control’ is a subset of a VCS

Distributed version control

No central server.
Every developer is a client, the server and the repositor.

This allows each developer to work independently and make changes to their local copy of the repository without requiring constant access to a central server. When changes are ready to be shared with the rest of the team, the developer can push their changes to the remote repository, which will then be available to other developers in the team.



Git distributed version control

No need to connect to central server

Can work without internet connection

No single failure point

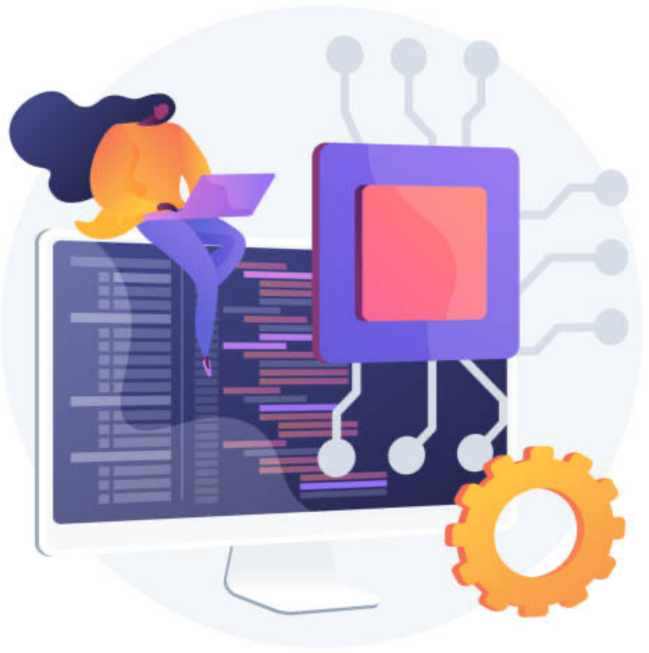
Developers can work independently and merge their work later

Every copy of a Git repository an serve either as the server or as a client (and has a complete history!

Git tracks **changes** not versions

Install Git: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

What is a repository?

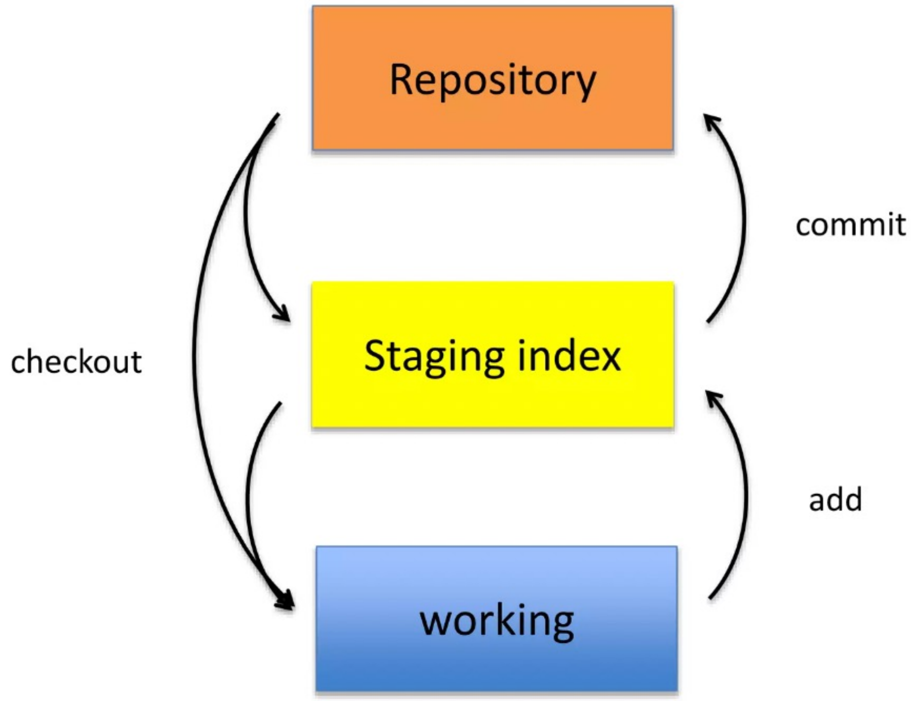


repo = repository

Usually used to organize a single project

Repos can contain folders and files, images, videos, spreadsheets, and datasets – anything your project needs

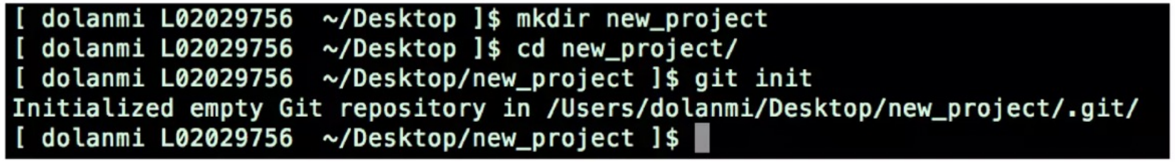
Git uses a three-tree architecture



A simple Git workflow

1. Initialize a new project in a directory:

   git init



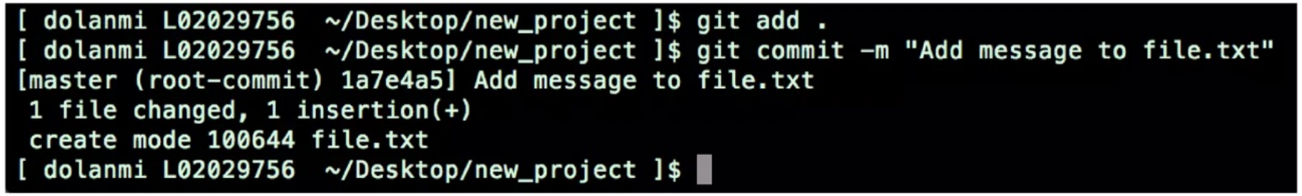
```
[ dolanmi L02029756  ~/Desktop ]$ mkdir new_project
[ dolanmi L02029756  ~/Desktop ]$ cd new_project/
[ dolanmi L02029756  ~/Desktop/new_project ]$ git init
Initialized empty Git repository in /Users/dolanmi/Desktop/new_project/.git/
[ dolanmi L02029756  ~/Desktop/new_project ]$ 
```

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

   git add .

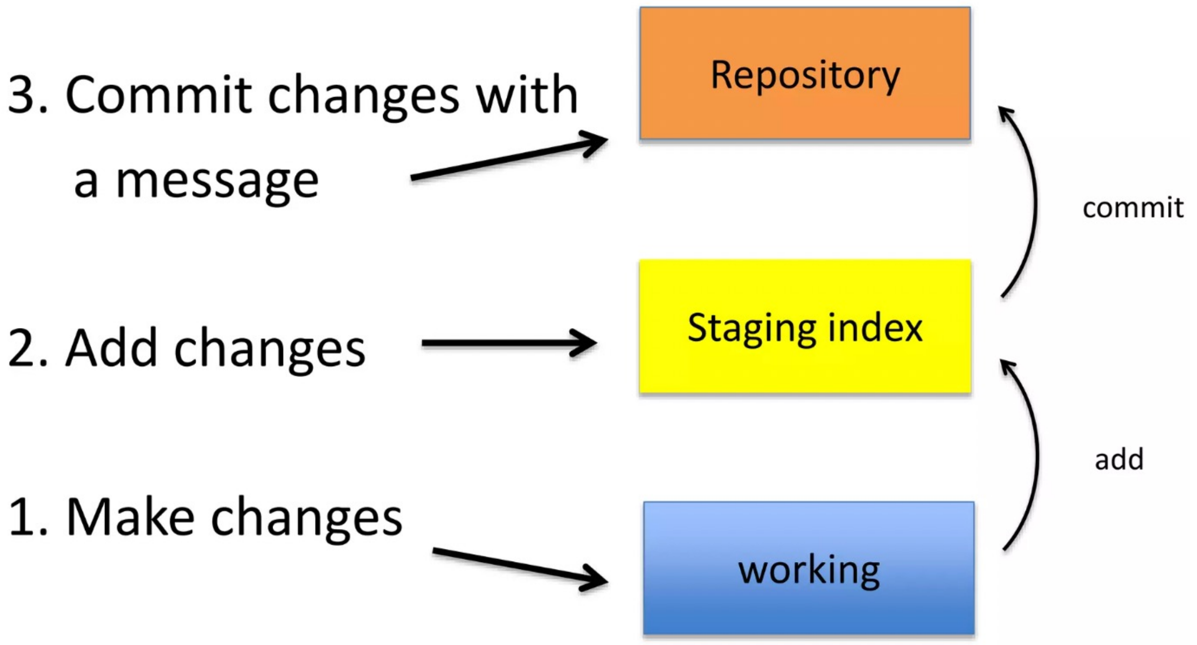
4. Commit the change to the repo:

   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

01

Tell what it does (present tense)

02

Single line summary followed by blank space followed by more complete description

03

Keep lines to <= 72 characters

04

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 Darrel”

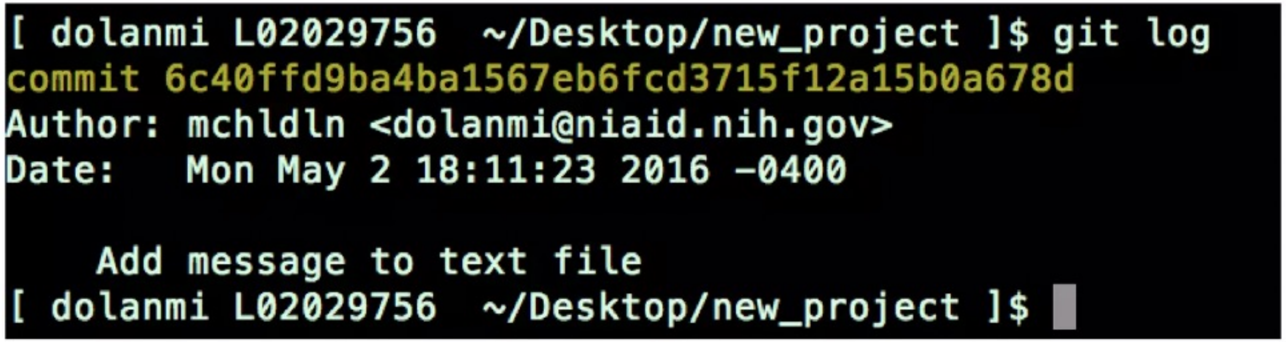
Bad: git commit -m "Fix login bug”

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 do I see what was done?

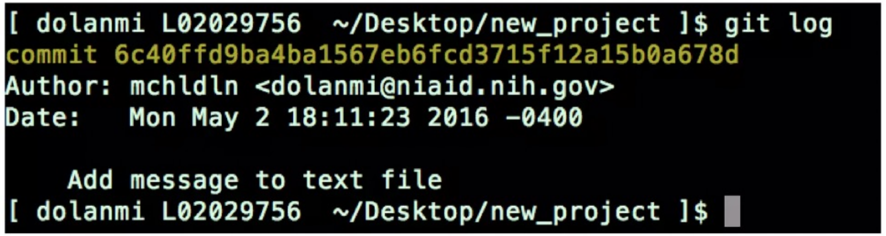
git log



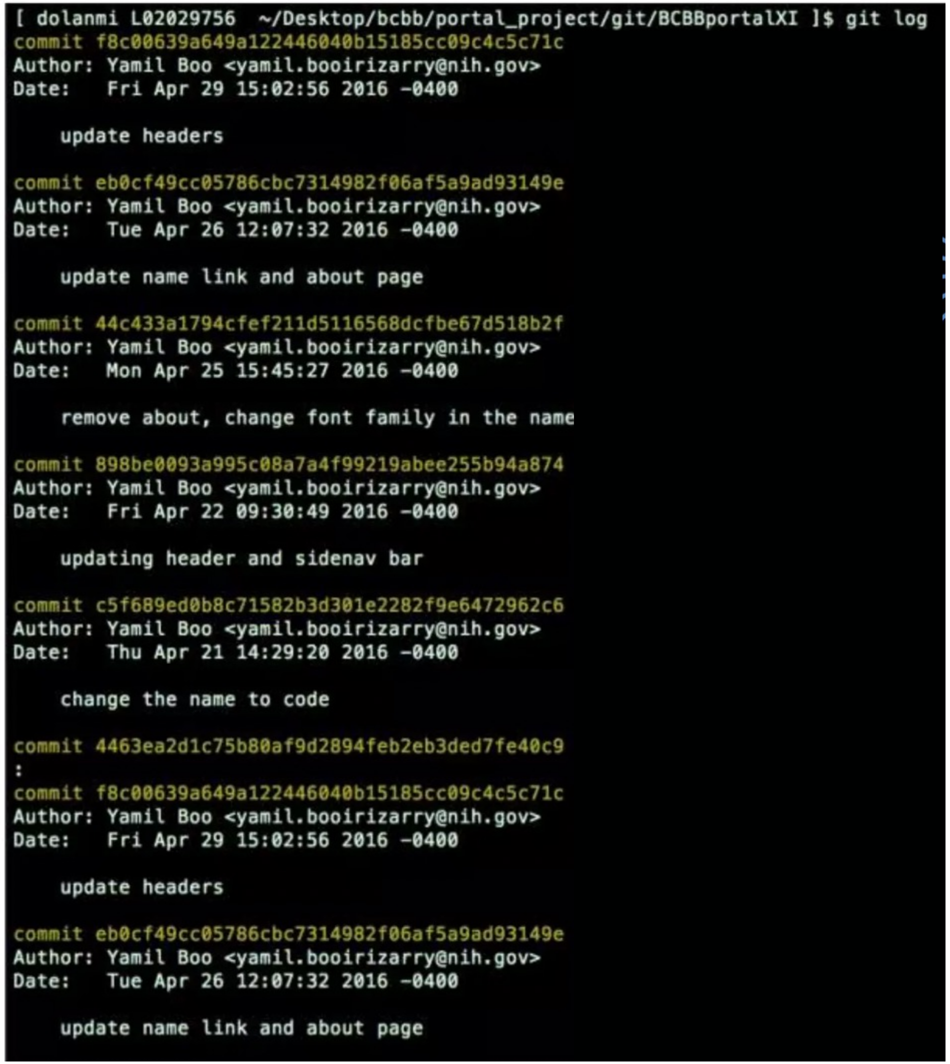
```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git log
commit 6c40ffd9ba4ba1567eb6fcd3715f12a15b0a678d
Author: mchldln <dolanmi@niaid.nih.gov>
Date:   Mon May 2 18:11:23 2016 -0400

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

How do I see what was done?

git log



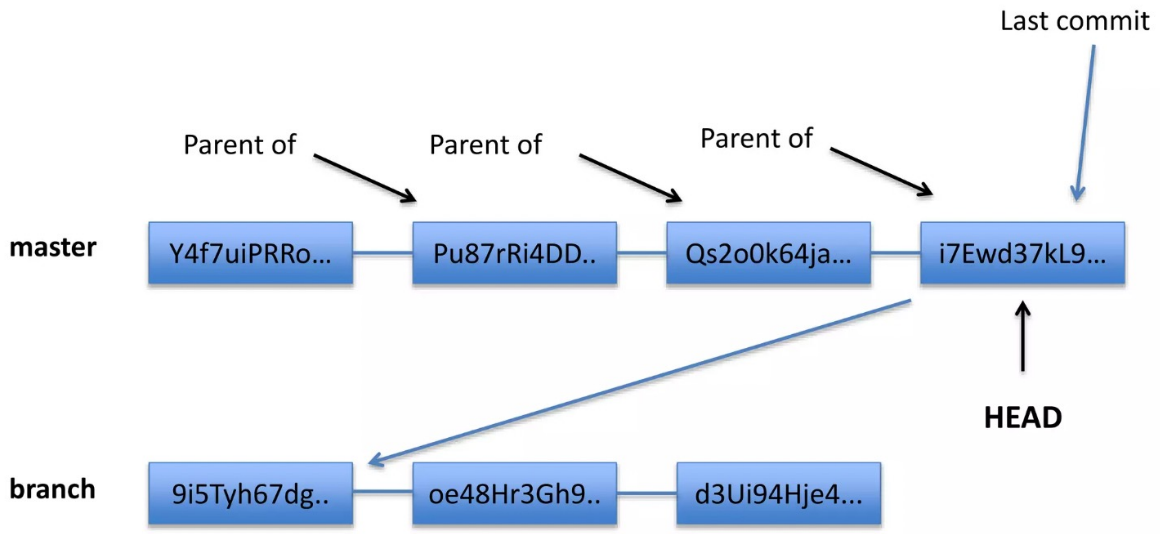
```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git log
commit 6c40ffd9ba4ba1567eb6fcd3715f12a15b0a678d
Author: mchldln <dolanmi@niaid.nih.gov>
Date:   Mon May 2 18:11:23 2016 -0400

    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
```

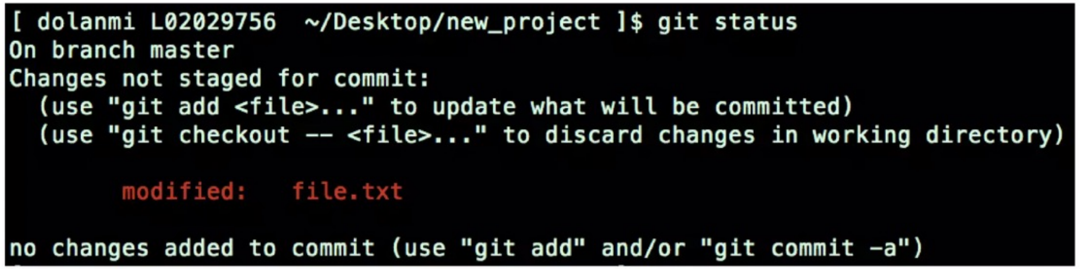
The HEAD pointer

- 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**
- last state of repo (what was checked out initially)
- HEAD points to parent of next commit (where writing the next commit takes place)

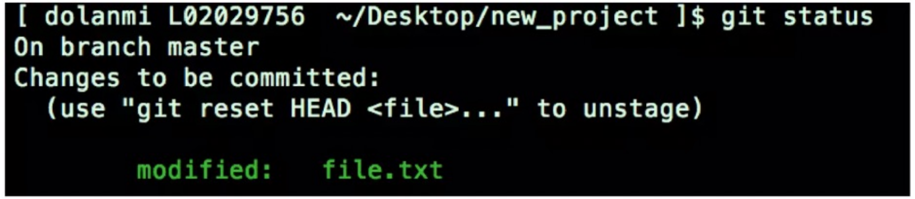


Which files were changed and where do they sit in the three tree?

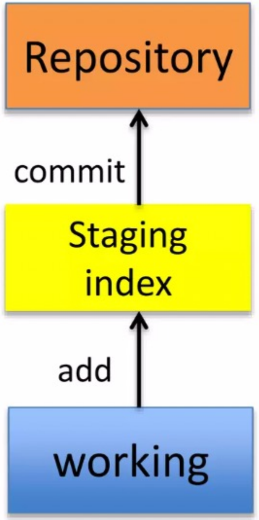
git status – allows one to see where files are in the three tree scheme



```
[ 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")
```

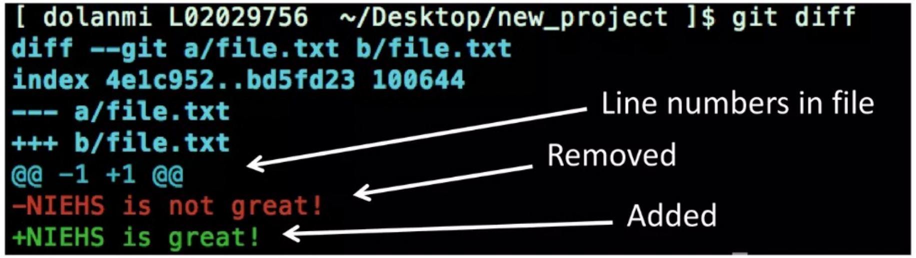


```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file.txt
```

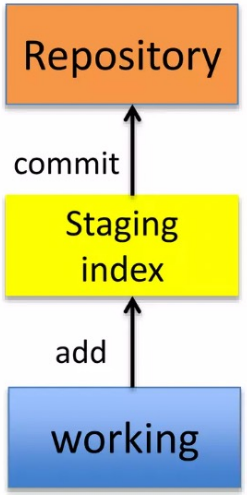


What changed in working directory?

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!
```



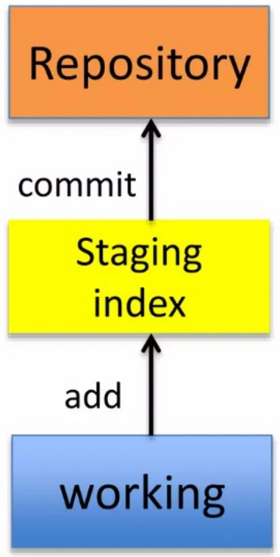
Note: git diff --staged - compares staging index to repo

Note: git diff filename can be used as well

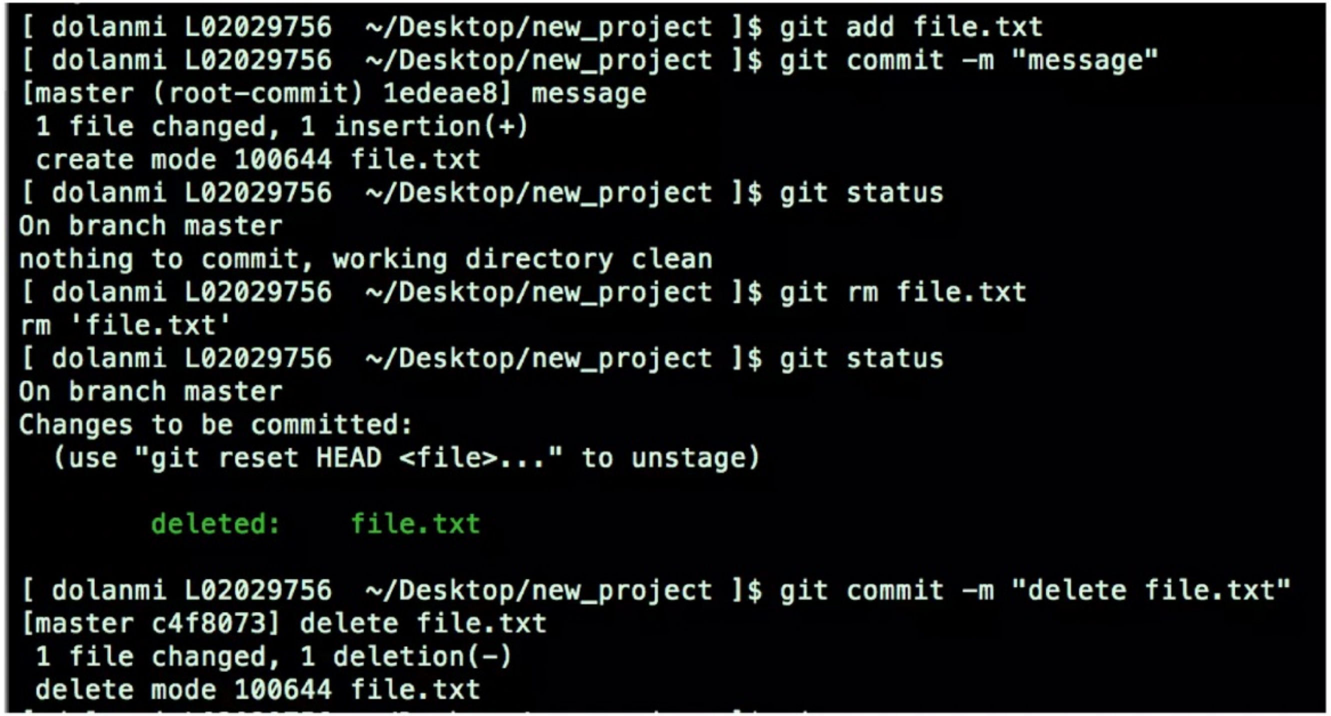
Deleting files from repo

```
git rm filename.txt
```

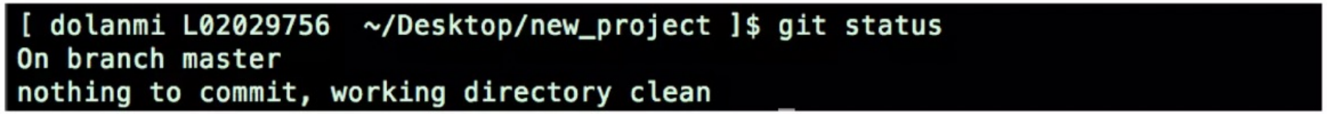
- moves deleted file change to staging area
- It is not enough to delete the file in your working directory. You must commit the change.



Deleting files from repo



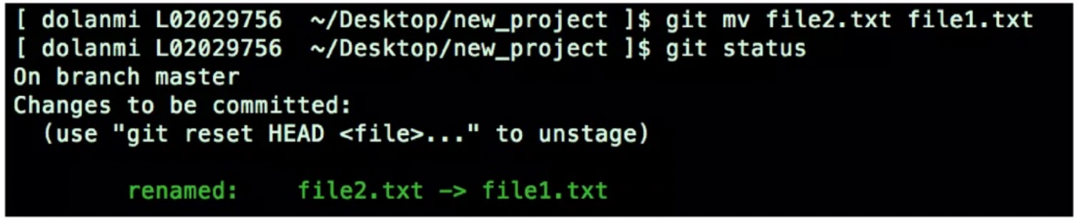
```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git add file.txt
[ dolanmi L02029756  ~/Desktop/new_project ]$ git commit -m "message"
[master (root-commit) 1edeae8] message
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch 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
```

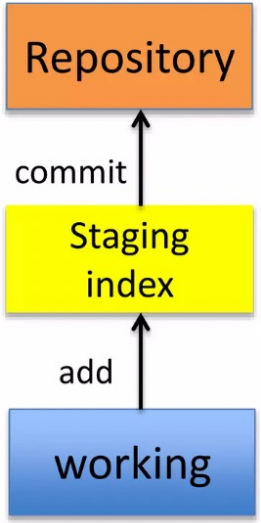
Moving (renaming files)

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

75% of the time you'll be using only these commands

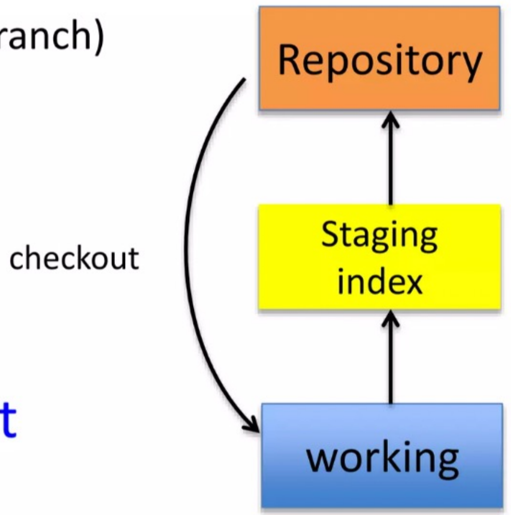
What if I want to undo changes made to working directory?

git checkout *something*

(where “something” is a file or an entire branch)

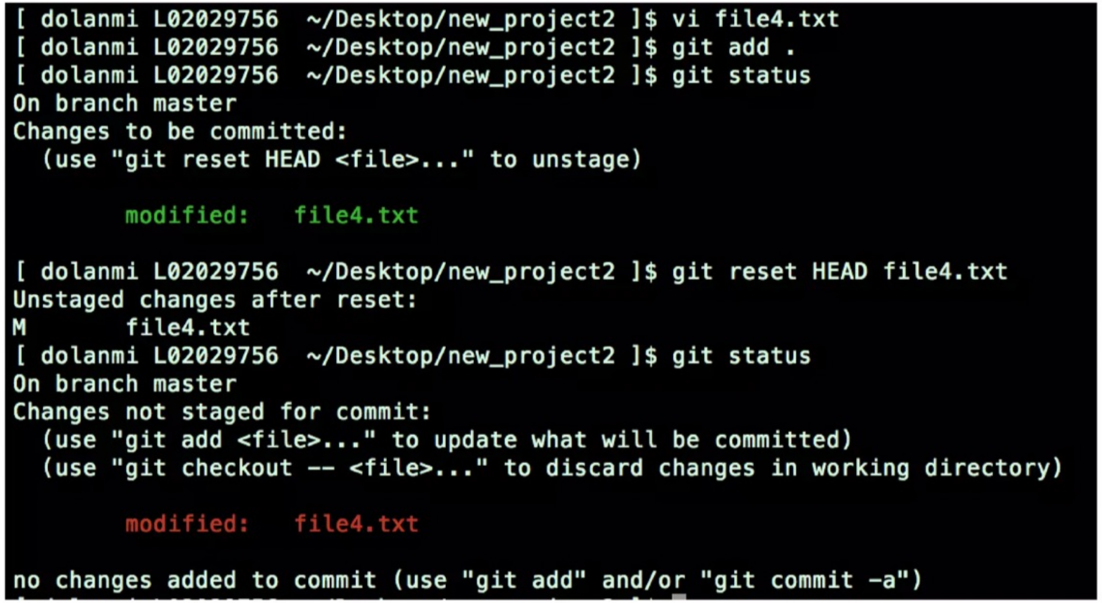
- git checkout will grab the file from the repo
- *Example:* git checkout -- file1.txt

(“checkout file ‘file1.txt’ from the current branch”)

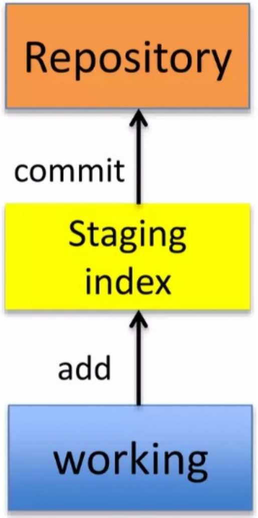


What if I want to undo changes added to staging area?

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")
```

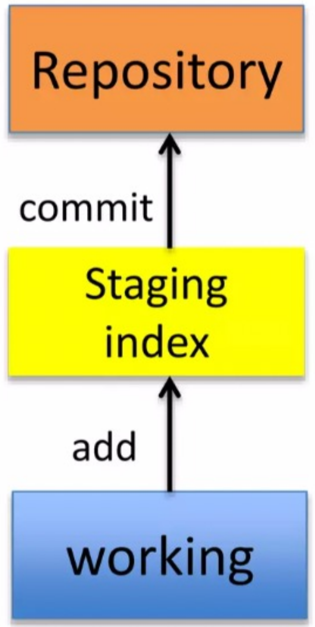


What if I want to undo changes added commited to the repo?

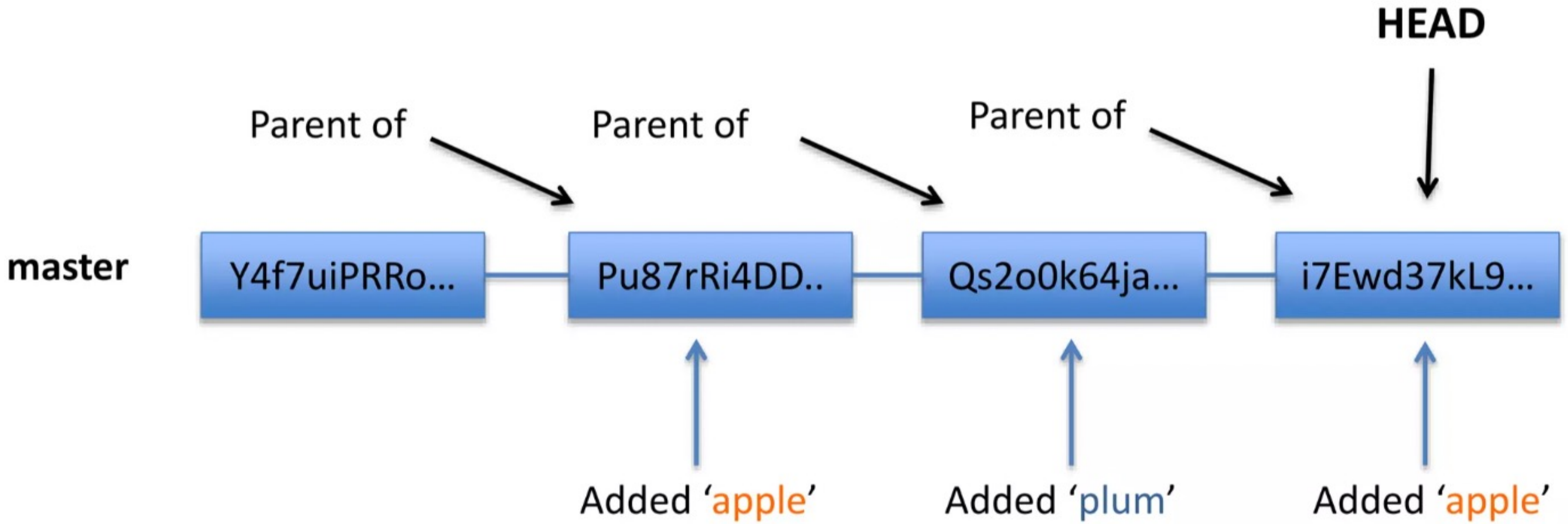
```
git commit --amend -m “message”
```

- allows one to amend a change to the last commit

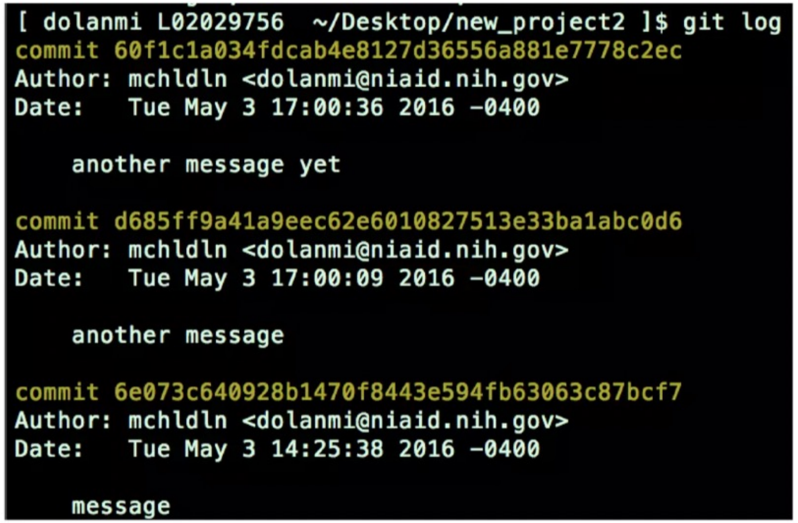
- anything in staging area will be amended to the last commit



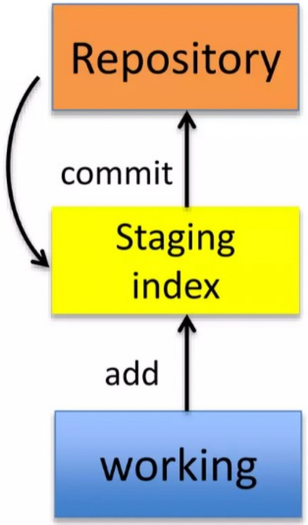
Note: To undo changes to older commits, make a new commit



Obtain older versions



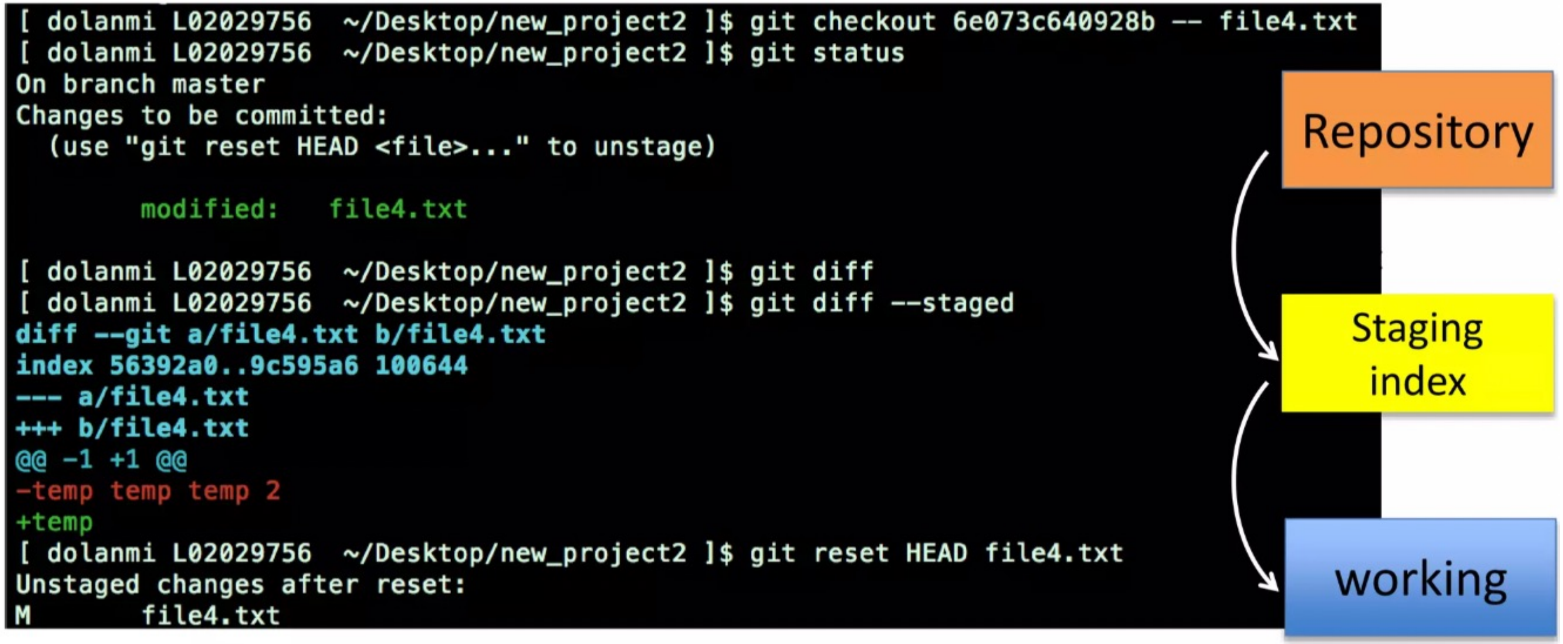
```
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git log
commit 60f1c1a034fdcab4e8127d36556a881e7778c2ec
Author: mchldln <dolanmi@niaid.nih.gov>
Date:   Tue May 3 17:00:36 2016 -0400

    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



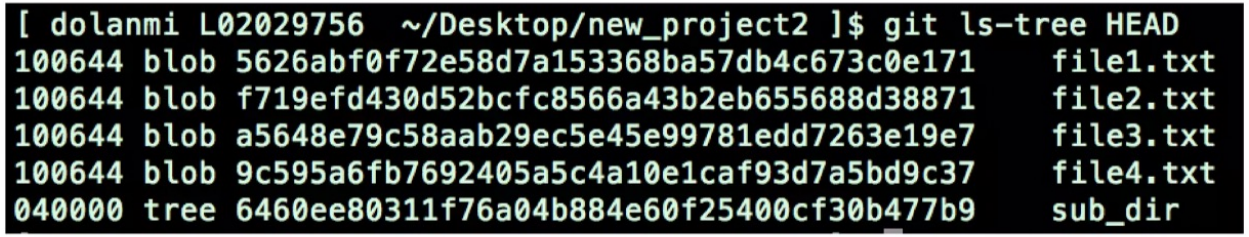
```
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git checkout 6e073c640928b -- file4.txt
[ dolanmi L02029756  ~/Desktop/new_project2 ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        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

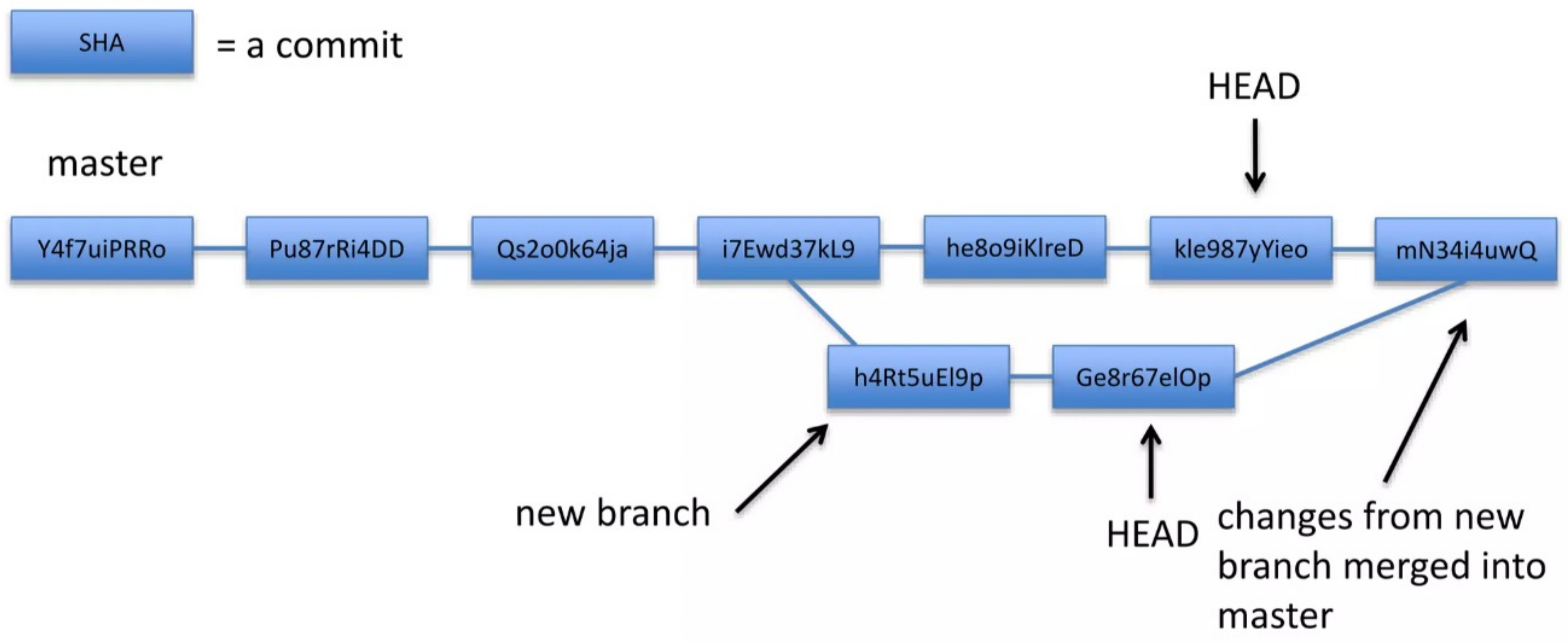
Allows one to try new ideas

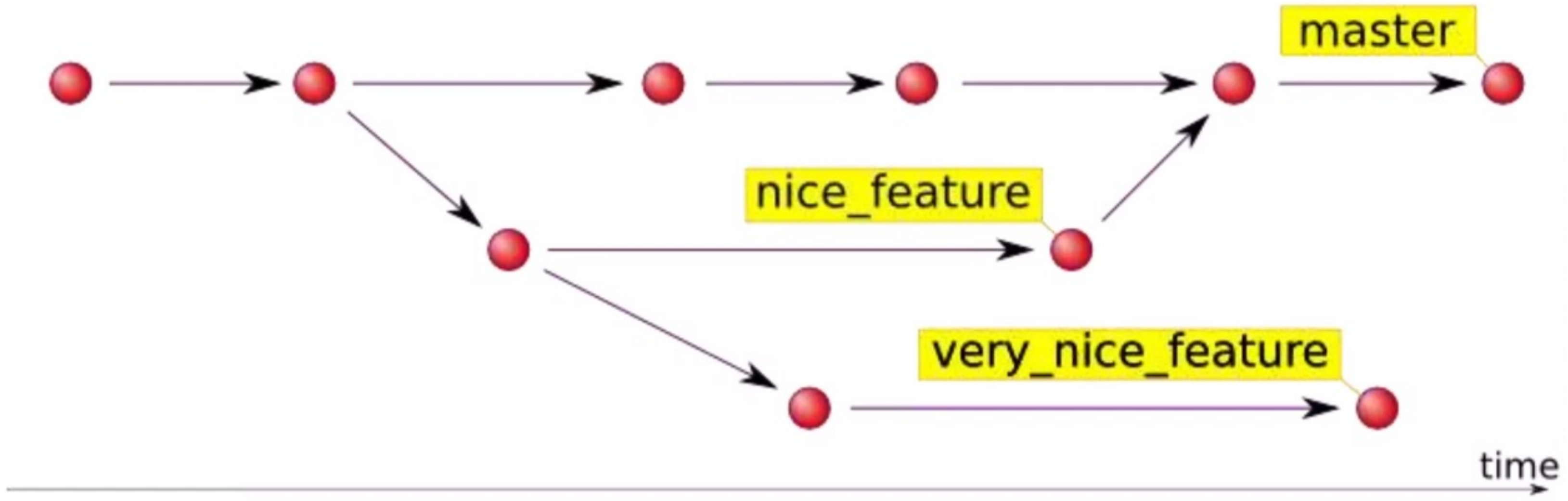
If an idea doesn't work, throw away the branch. Don't have to undo many changes to master branch

If it does work, merge ideas into marter branch

There is only one working directory

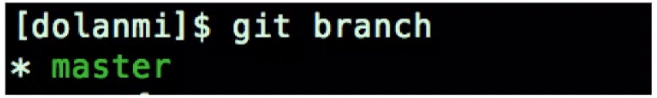
Branching and merging example




master
nice_feature
very_nice_feature
time

In which branch am I?

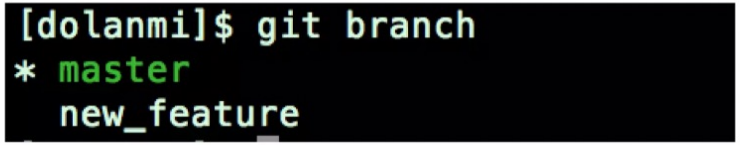
git branch



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

How do I create a new branch?

git branch new_branch_name

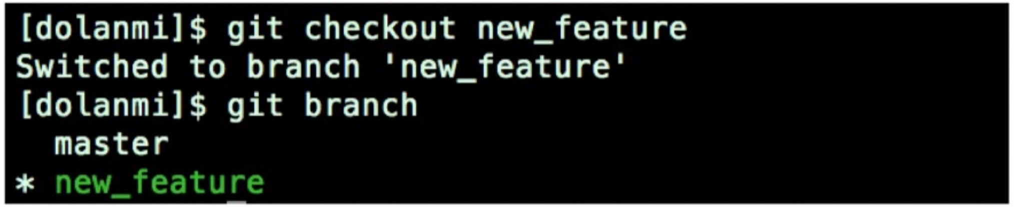


```
[dolanmi]$ git branch
* master
  new_feature
```

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



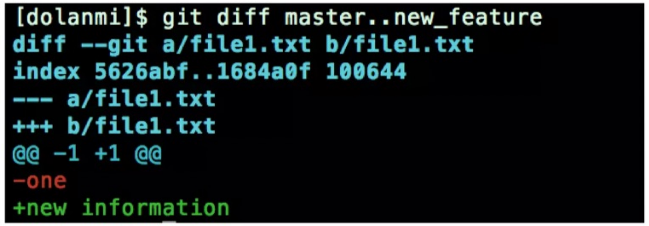
```
[dolanmi]$ git checkout new_feature
Switched to branch 'new_feature'
[dolanmi]$ git branch
  master
* new_feature
```

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

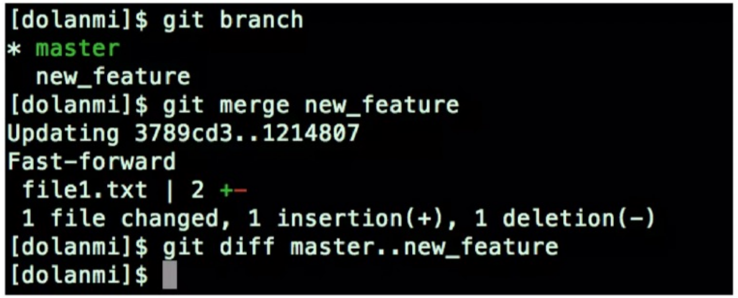


```
[dolanmi]$ git diff master..new_feature
diff --git a/file1.txt b/file1.txt
index 5626abf..1684a0f 100644
--- a/file1.txt
+++ b/file1.txt
@@ -1 +1 @@
-one
+new information
```

How do I merge a branch?

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

git merge branch_to_merge

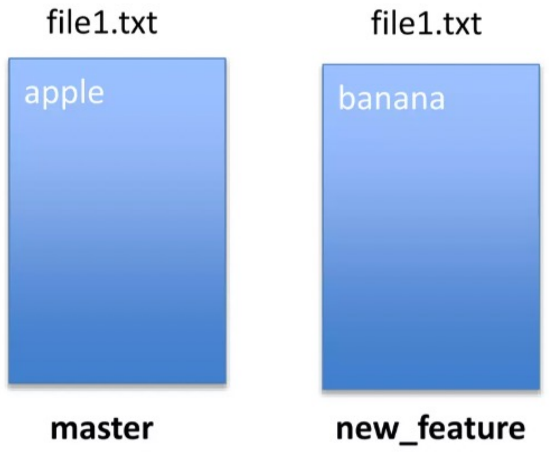


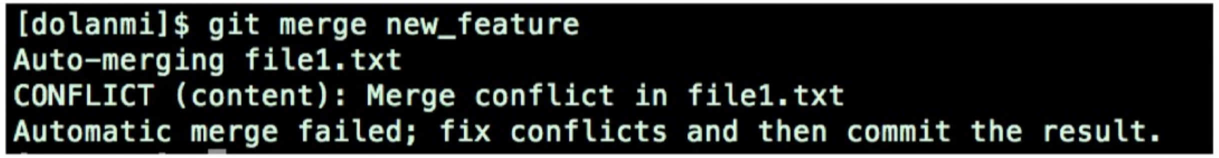
```
[dolanmi]$ git branch
* master
  new_feature
[dolanmi]$ git merge new_feature
Updating 3789cd3..1214807
Fast-forward
 file1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[dolanmi]$ git diff master..new_feature
[dolanmi]$
```

Note: Always have a clean working directory when merging

Merge conflits

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

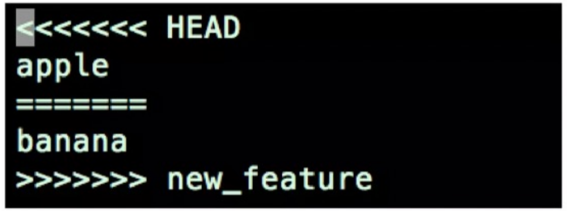




```
[dolanmi]$ git merge new_feature
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Resolving merge conflits

Git will notate the conflict in the files!

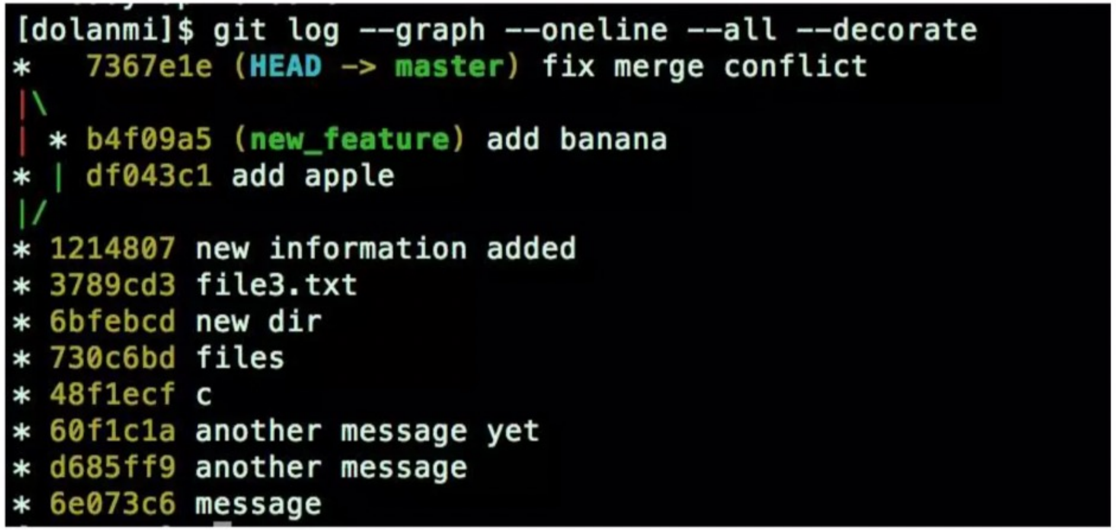


```
<<<<<<< HEAD
apple
=======
banana
>>>>>>> new_feature
```

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

1 Merge often

2 Keep commits small/focused

3 Bring changes occurring to master into your branch frequently (“tracking”)

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

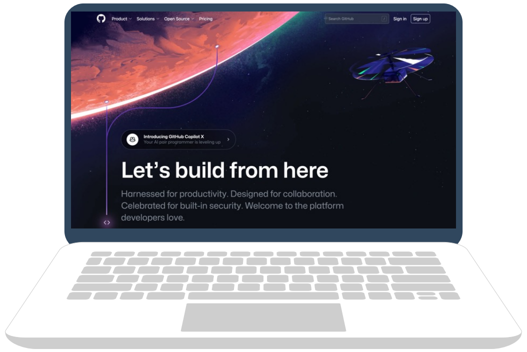
What is Github?

Github

A plataform to host git code repositories

http://github.com

Launched in 2008


Product
Solutions
Open Source
Pricing
Search GitHub
Sign in
Sign up
Introducing GitHub Copilot X
Your AI pair programmer is leveling up

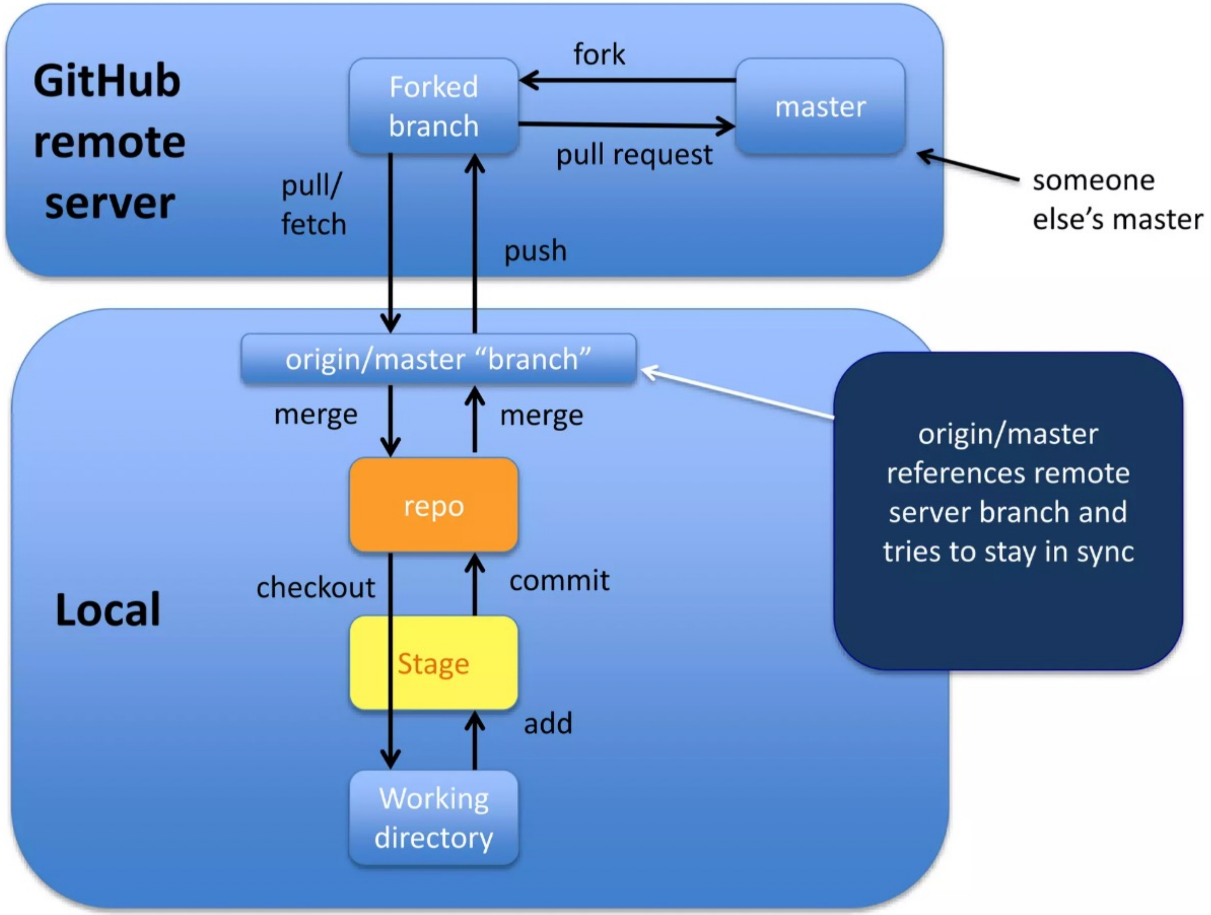
Let's build from here

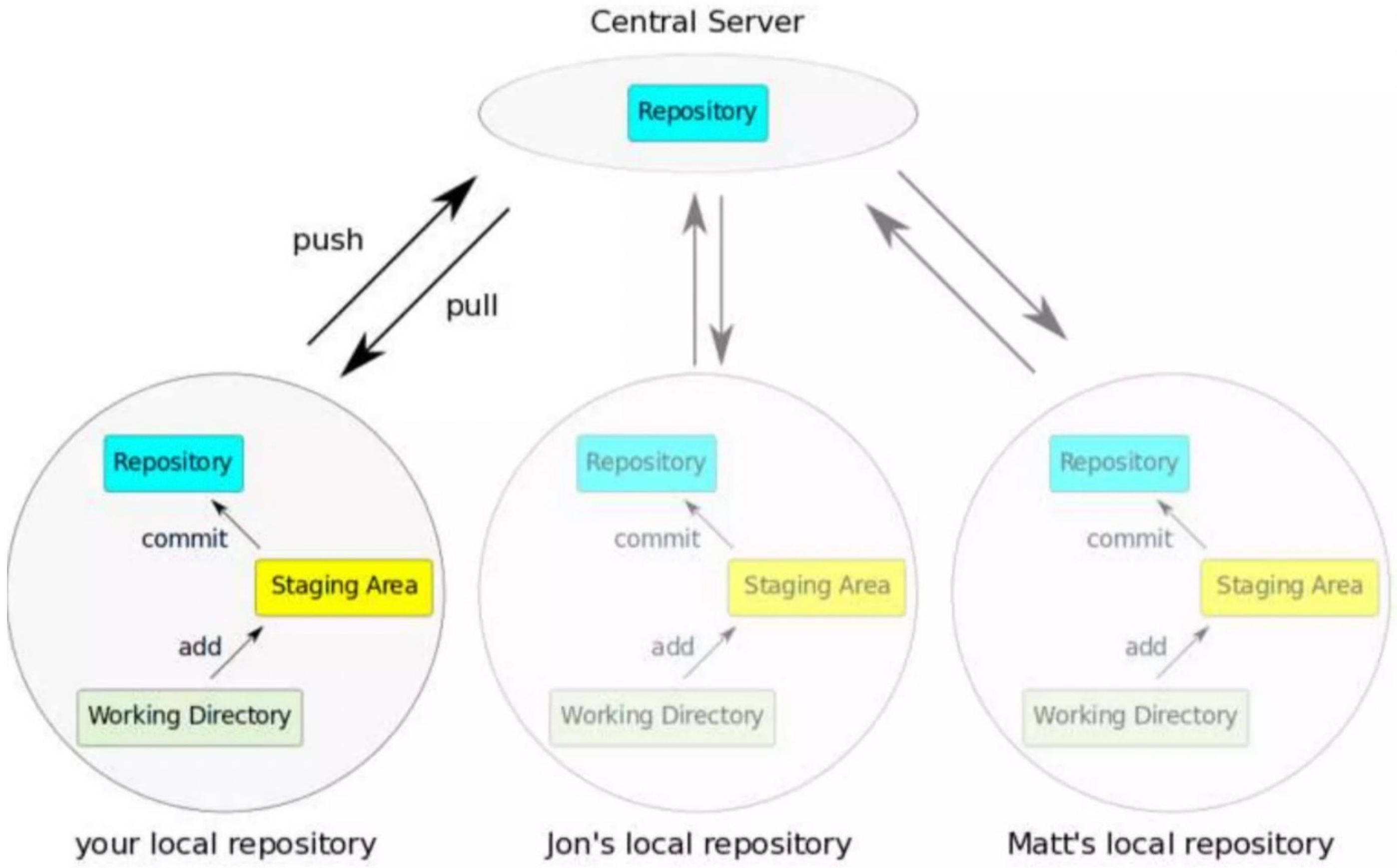
Harnessed for productivity. Designed for collaboration. Celebrated for built-in security. Welcome to the platform developers love.

Most popular Git host

Allows users to collaborate on projects from anywhere

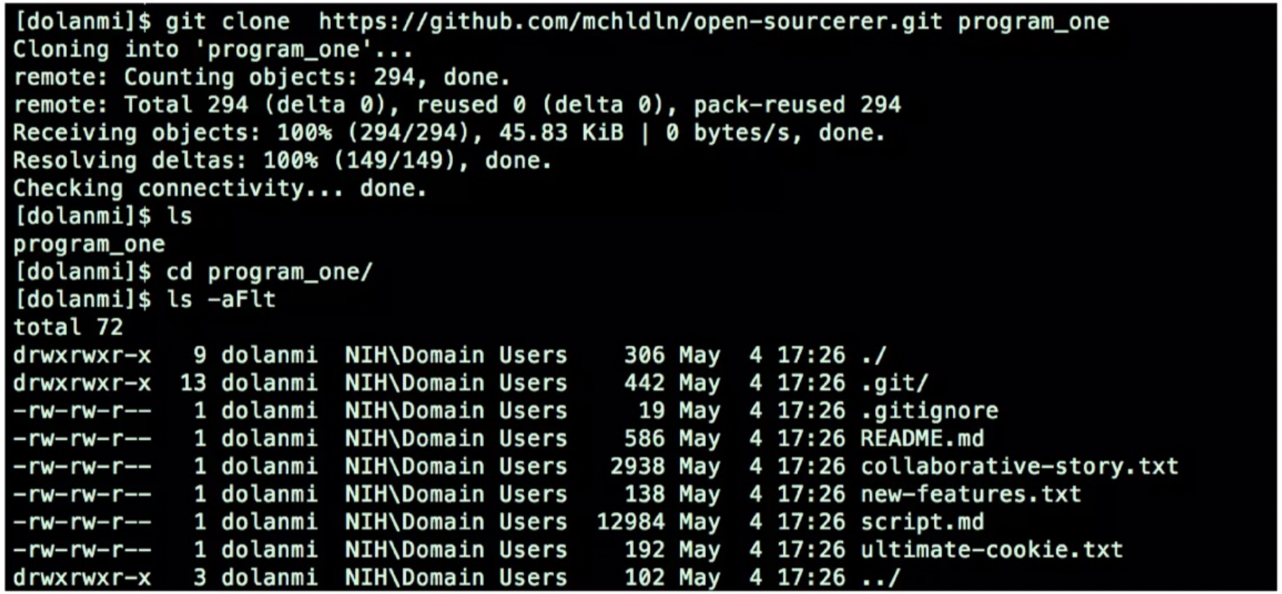
Github makes git social!


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

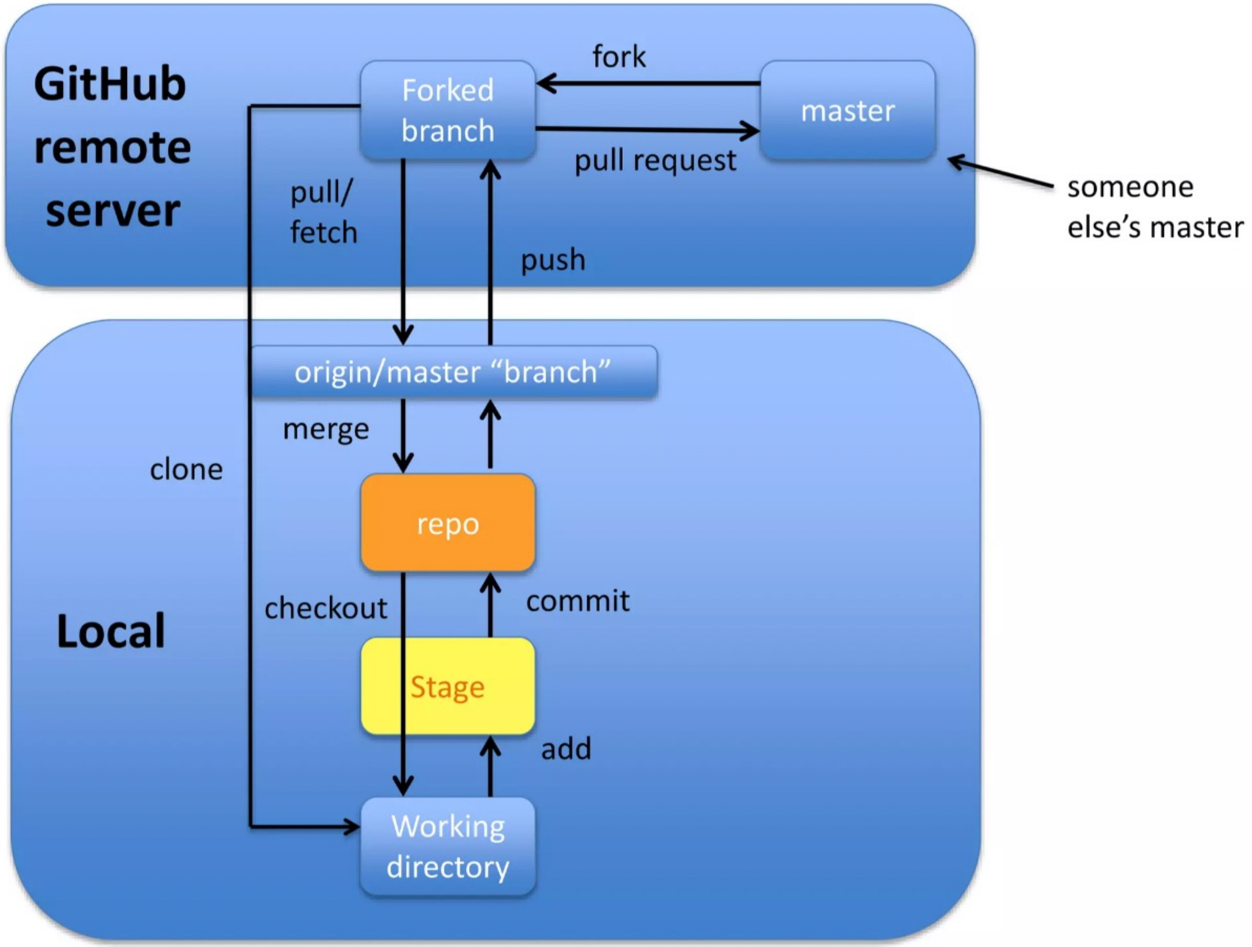

Central Server
Repository
push
pull
Repository
commit
Staging Area
add
Working Directory
Repository
commit
Staging Area
add
Working Directory
Repository
commit
Staging Area
add
Working Directory
your local repository
Jon's local repository
Matt's local repository

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

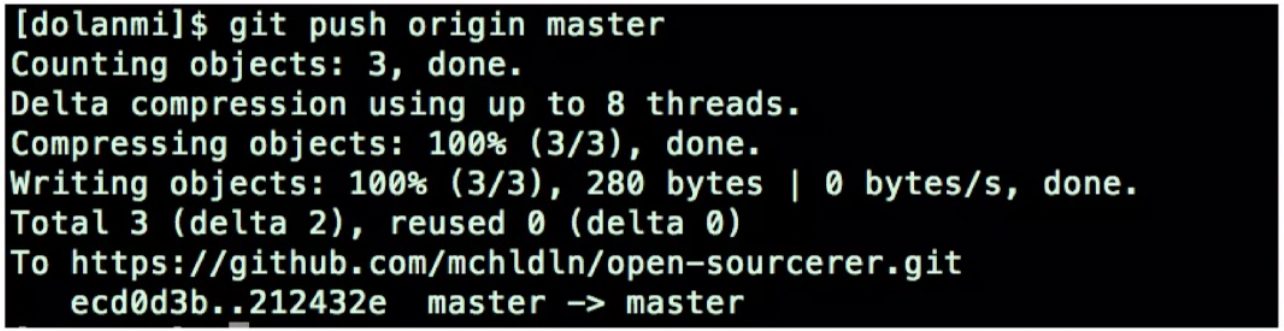
Note: Yes! You may have more than one remote linked to your local directory!

Which remotes am I linked to?

```
git remote
```

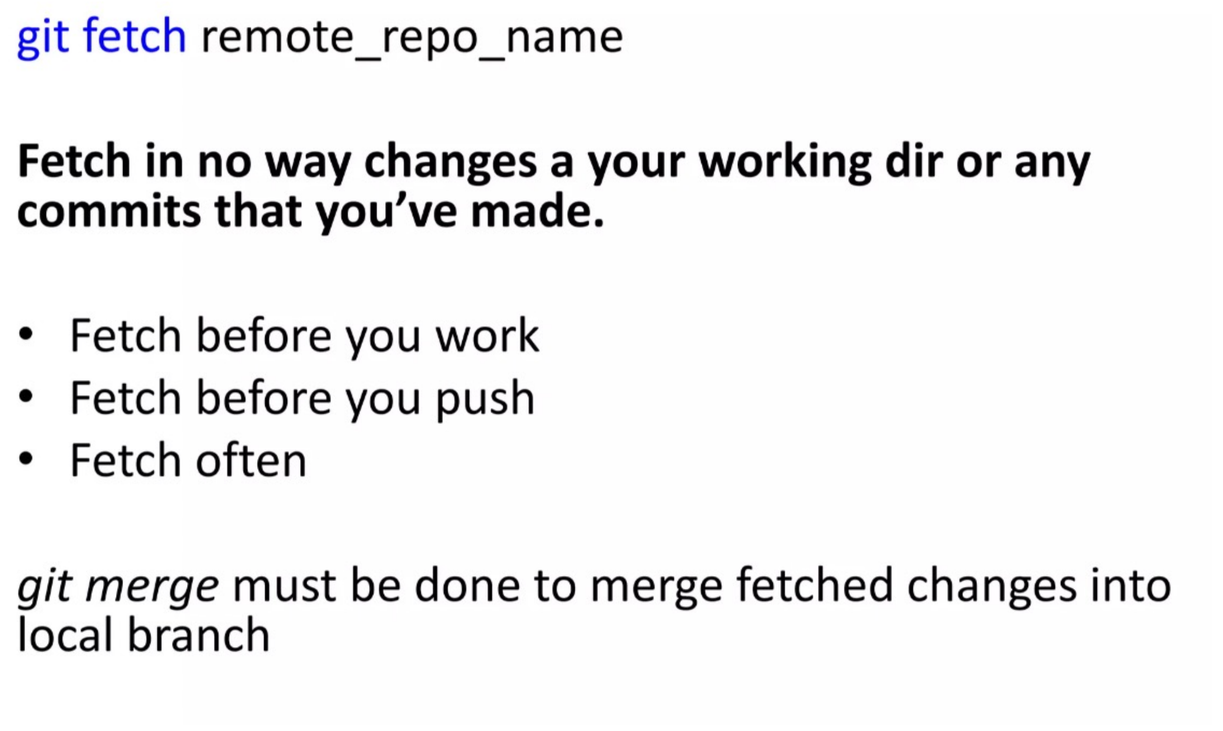
Pushing to a remote repo

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



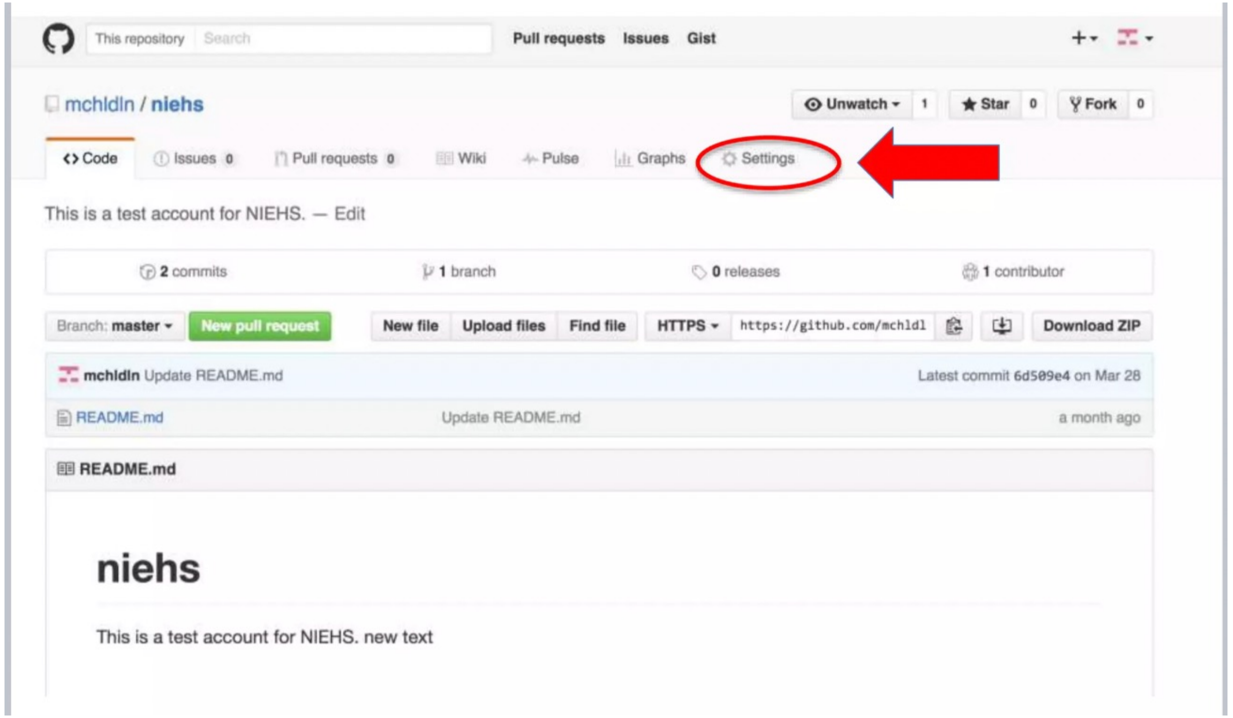
git fetch remote_repo_name

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 is a test account for NIEHS. — Edit

2 commits | 1 branch | 0 releases | 1 contributor

https://github.com/mchldl

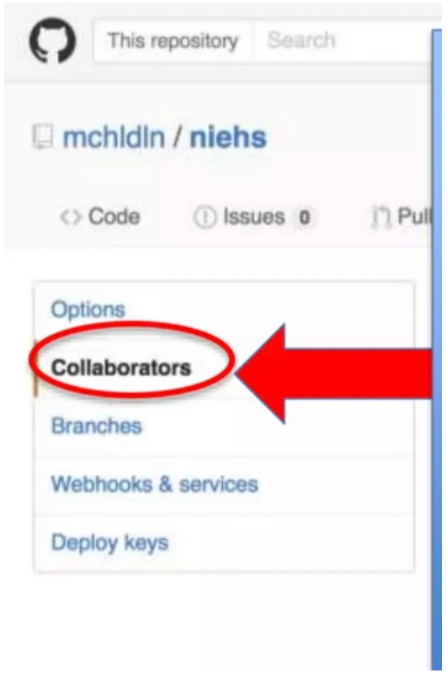
mchldln Update README.md — Latest commit 6d509e4 on Mar 28

README.md — Update README.md — a month ago

README.md

niehs

This is a test account for NIEHS. new text



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

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

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

APLICAÇÕES INFORMÁTICAS EM ENGENHARIA BIOMÉDICA