GitHub

Introduction to 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.

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

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

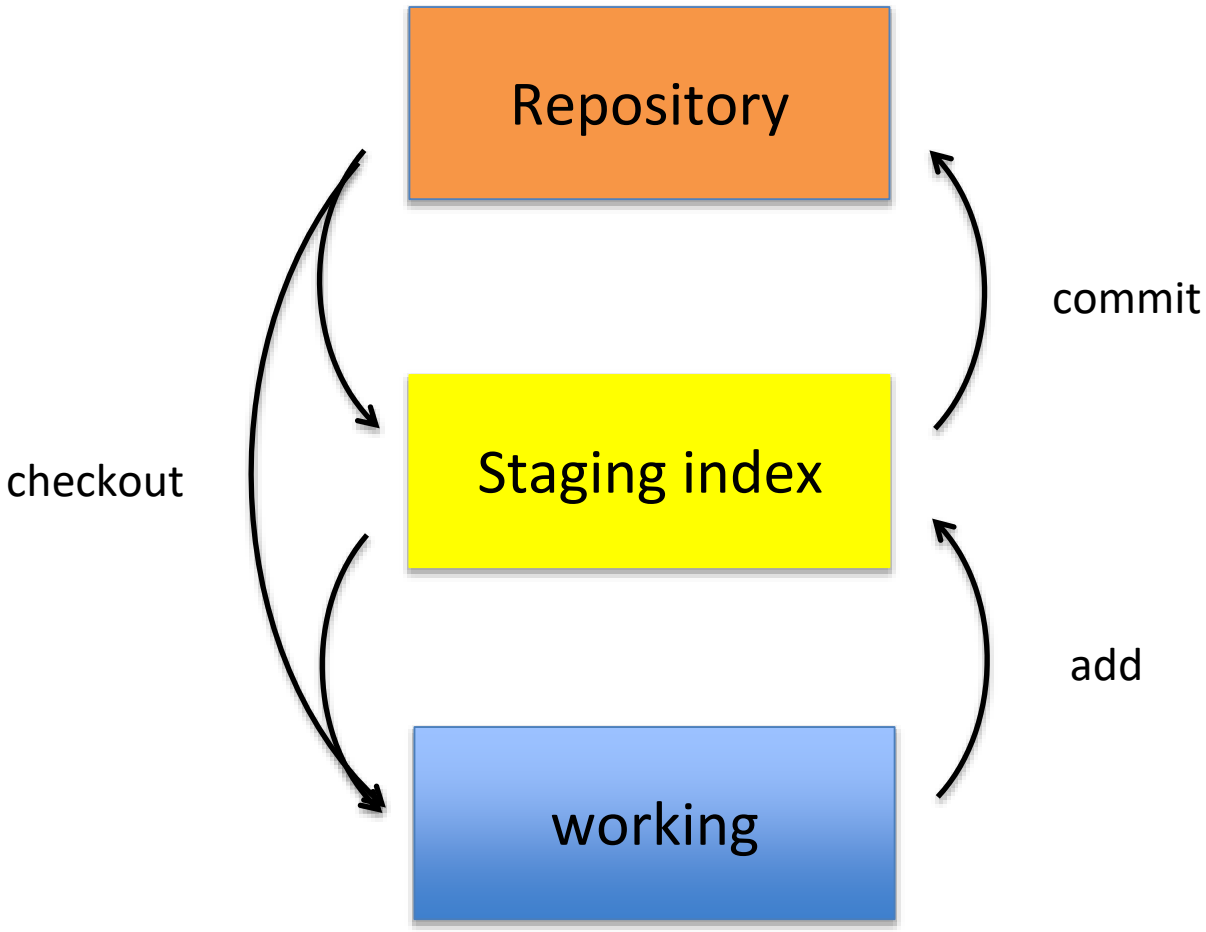
- Anyone wanting to track edits (especially changes to text files)
  - review history of changes
  - anyone wanting to share, merge changes

- Anyone not afraid of
  command line tools

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

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:

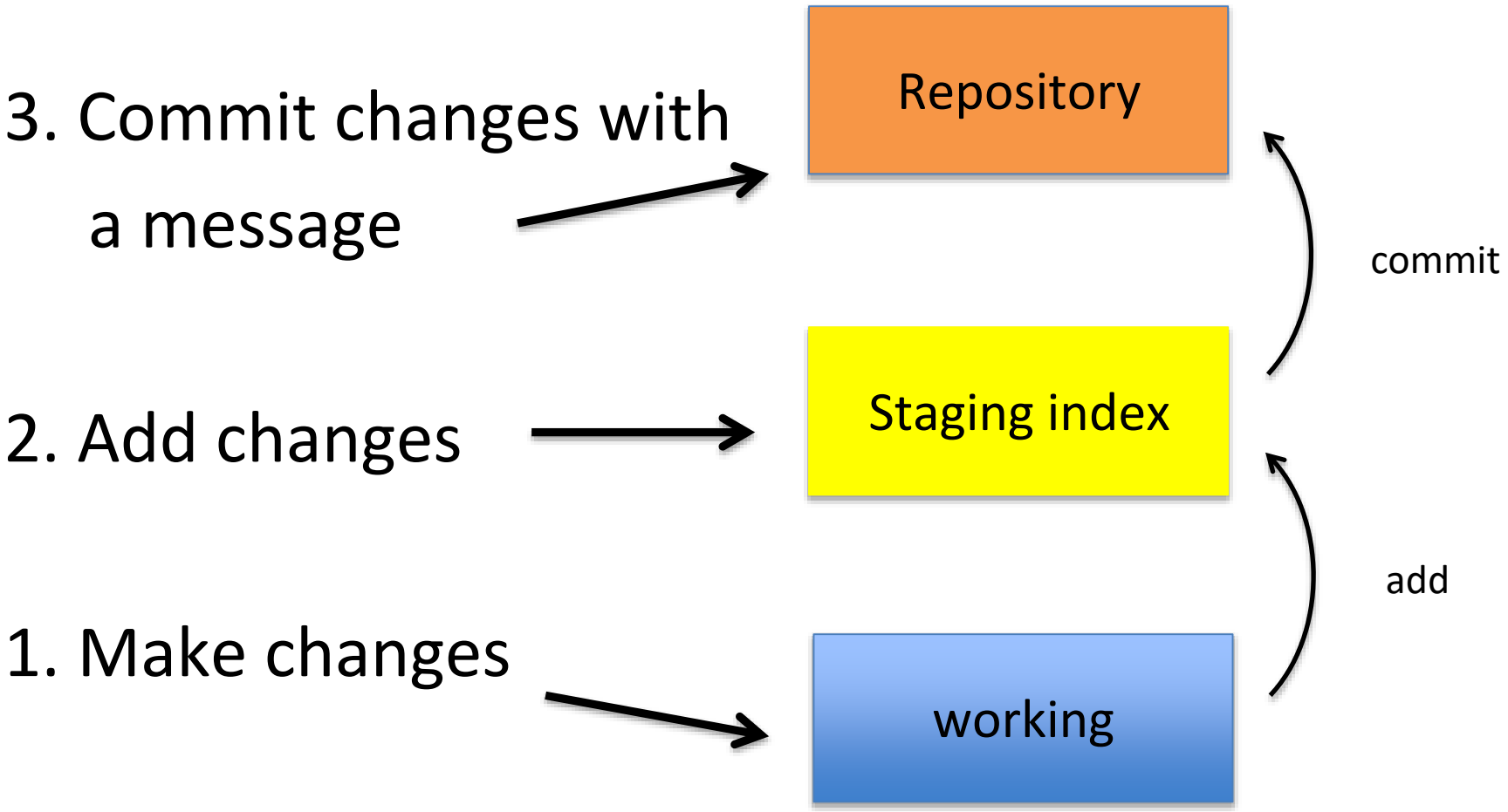
   git add .

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

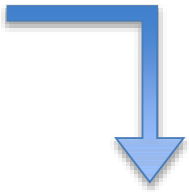
Bad: "Typo fix"

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

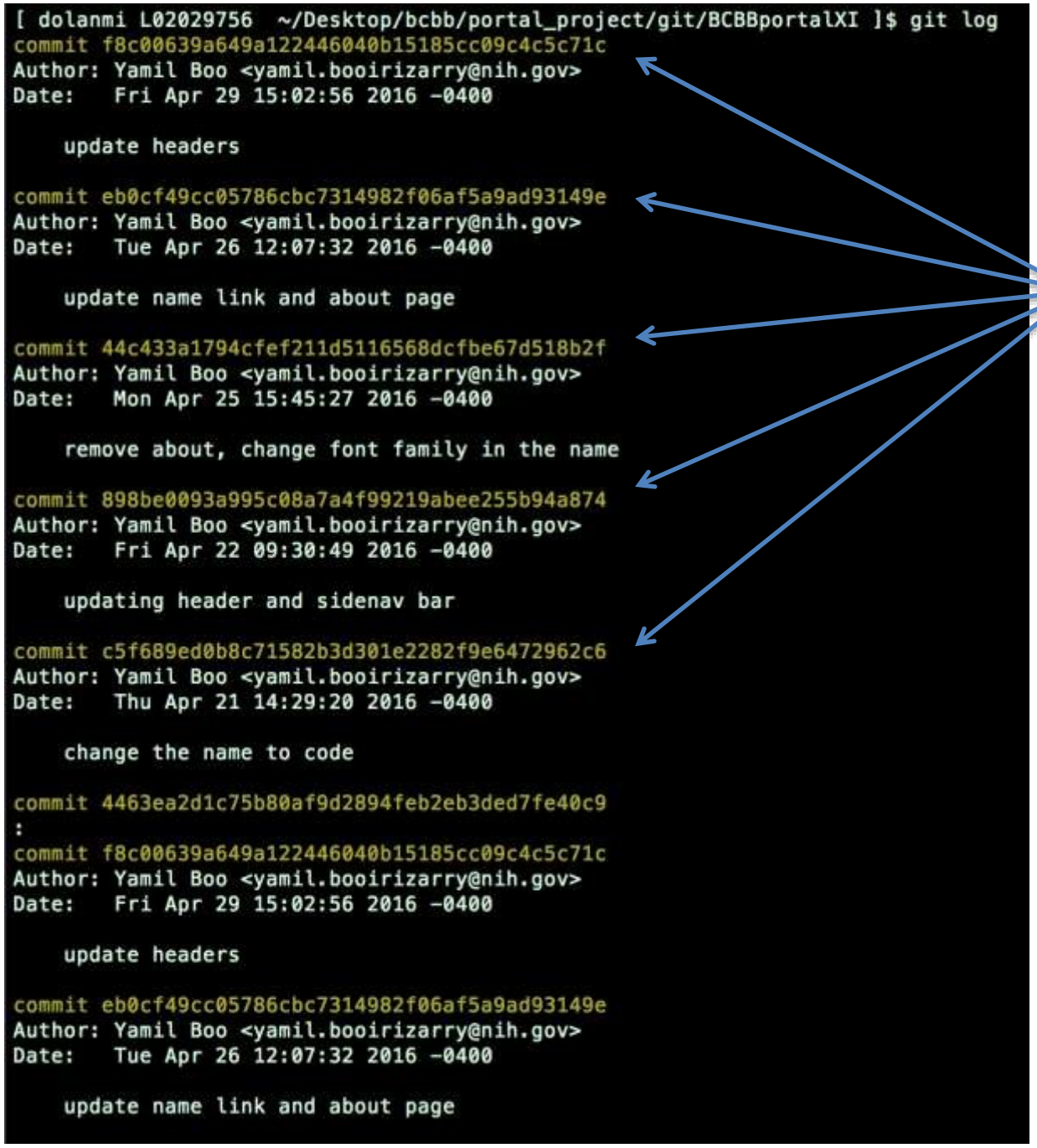


```
Redirect user to the requested page after login

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

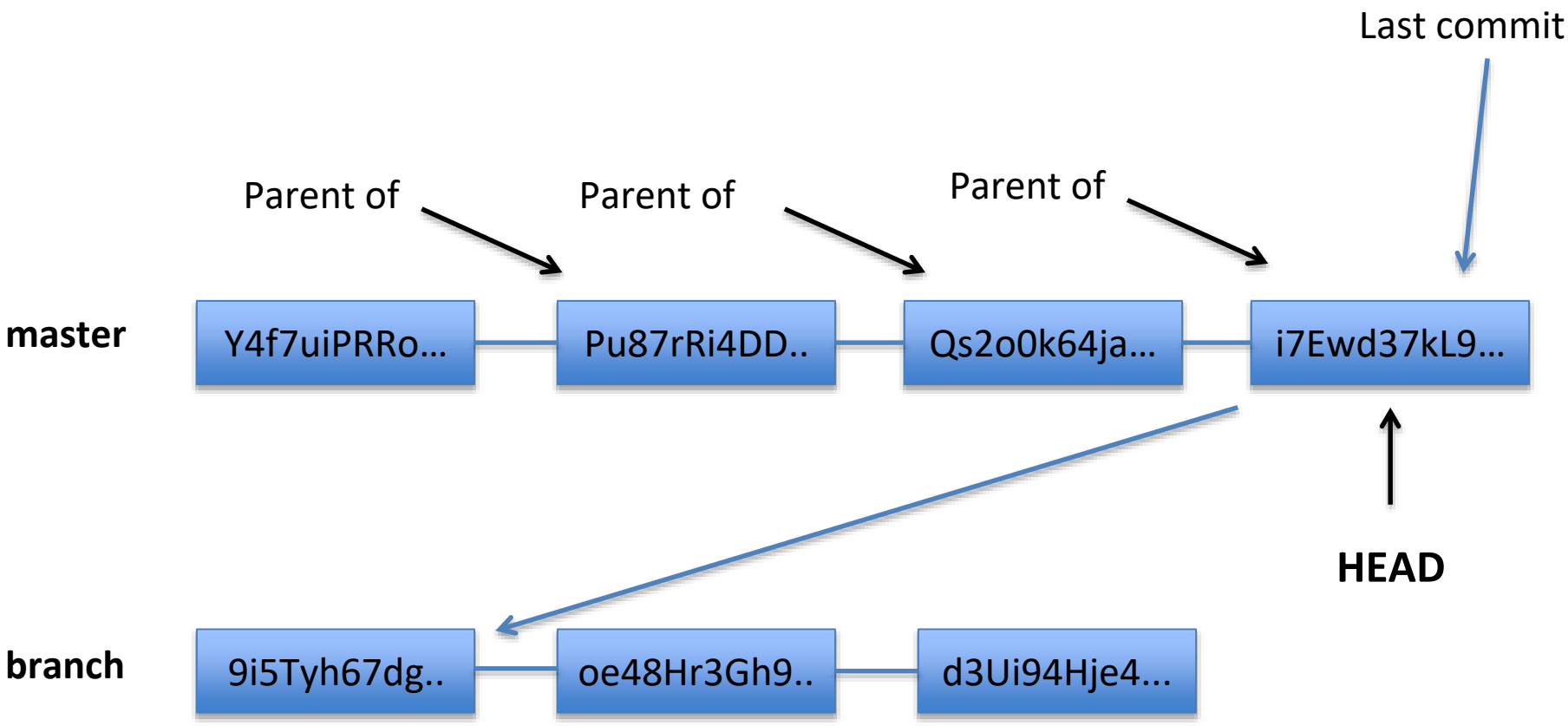
```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git log
commit 6c40ffd9ba4ba1567eb6fcd3715f12a15b0a678d
Author: mchldln <dolanmi@niaid.nih.gov>
Date:   Mon May 2 18:11:23 2016 -0400

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



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)

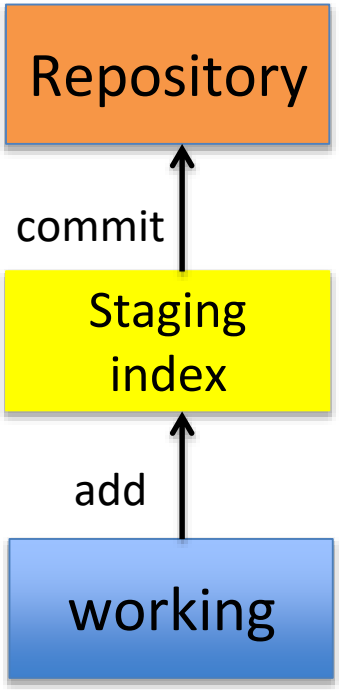

Last commit
Parent of
Parent of
Parent of
master
Y4f7uiPRRo…
Pu87rRi4DD..
Qs2o0k64ja…
i7Ewd37kL9…
HEAD
branch
9i5Tyh67dg..
oe48Hr3Gh9..
d3Ui94Hje4…

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?

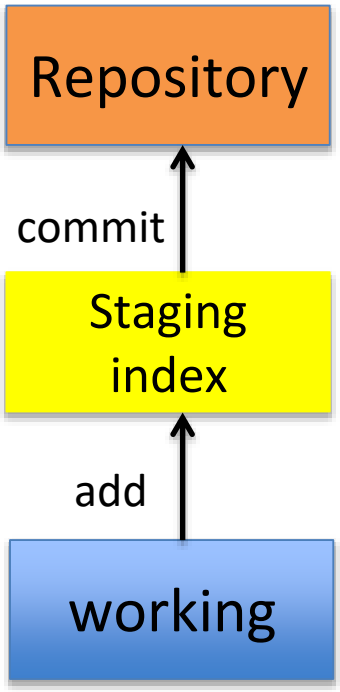
git diff – compares changes to files between repo and working directory

```
[ 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

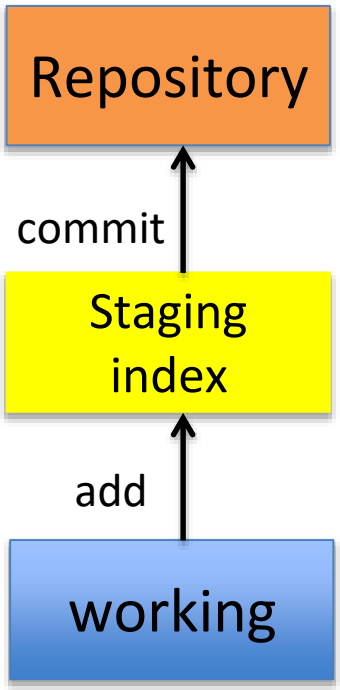
Note: git diff filename can be used as well

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
```

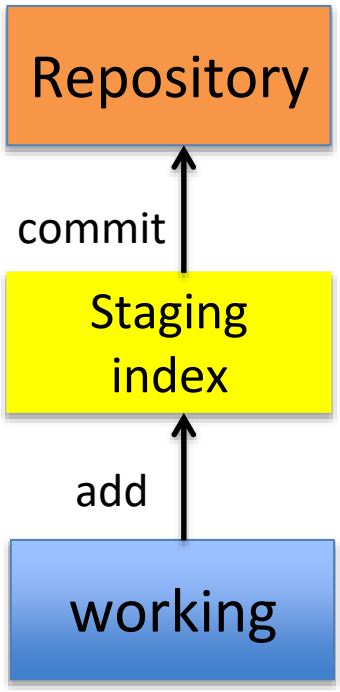
```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
nothing to commit, working directory clean
```

Moving (renaming) files

git mv filename1.txt filename2.txt

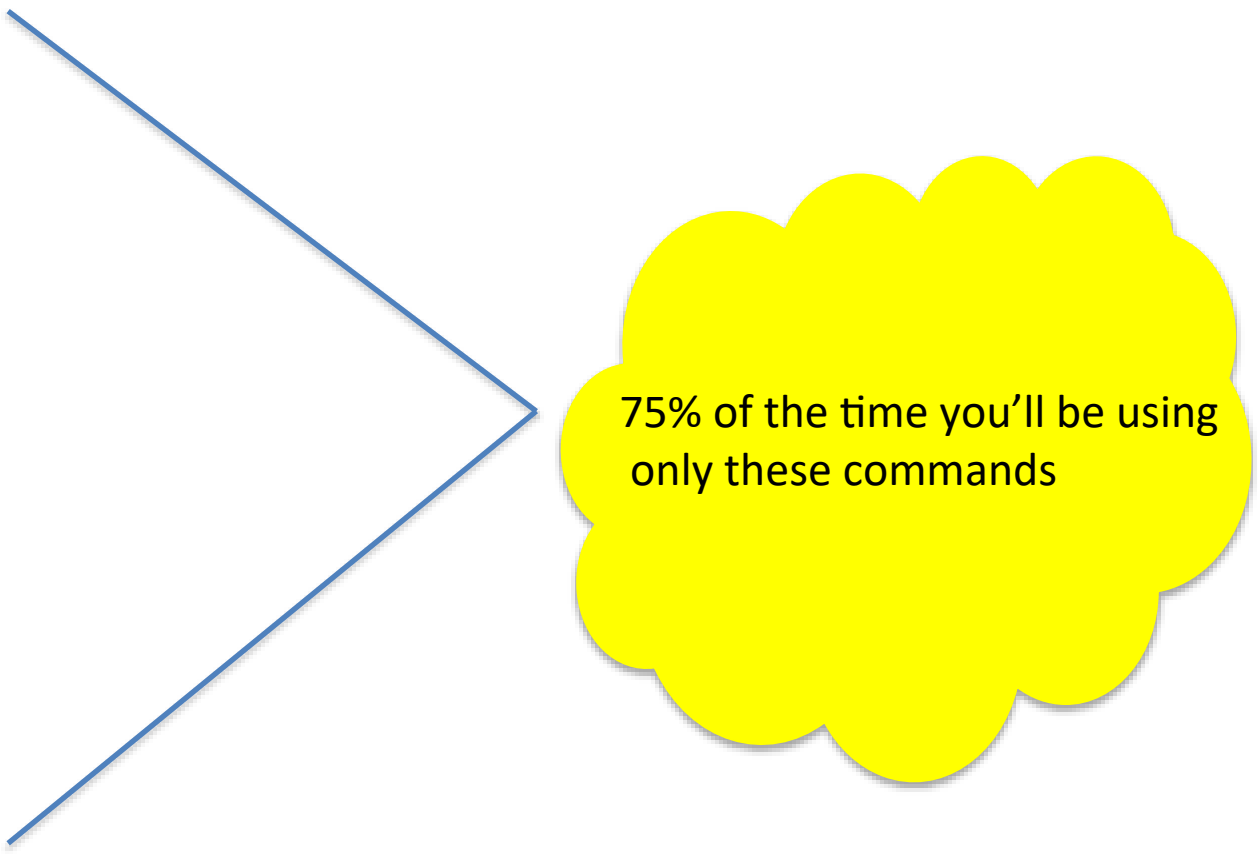
```
[ dolanmi L02029756  ~/Desktop/new_project ]$ git mv file2.txt file1.txt
[ dolanmi L02029756  ~/Desktop/new_project ]$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

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

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



Good news!

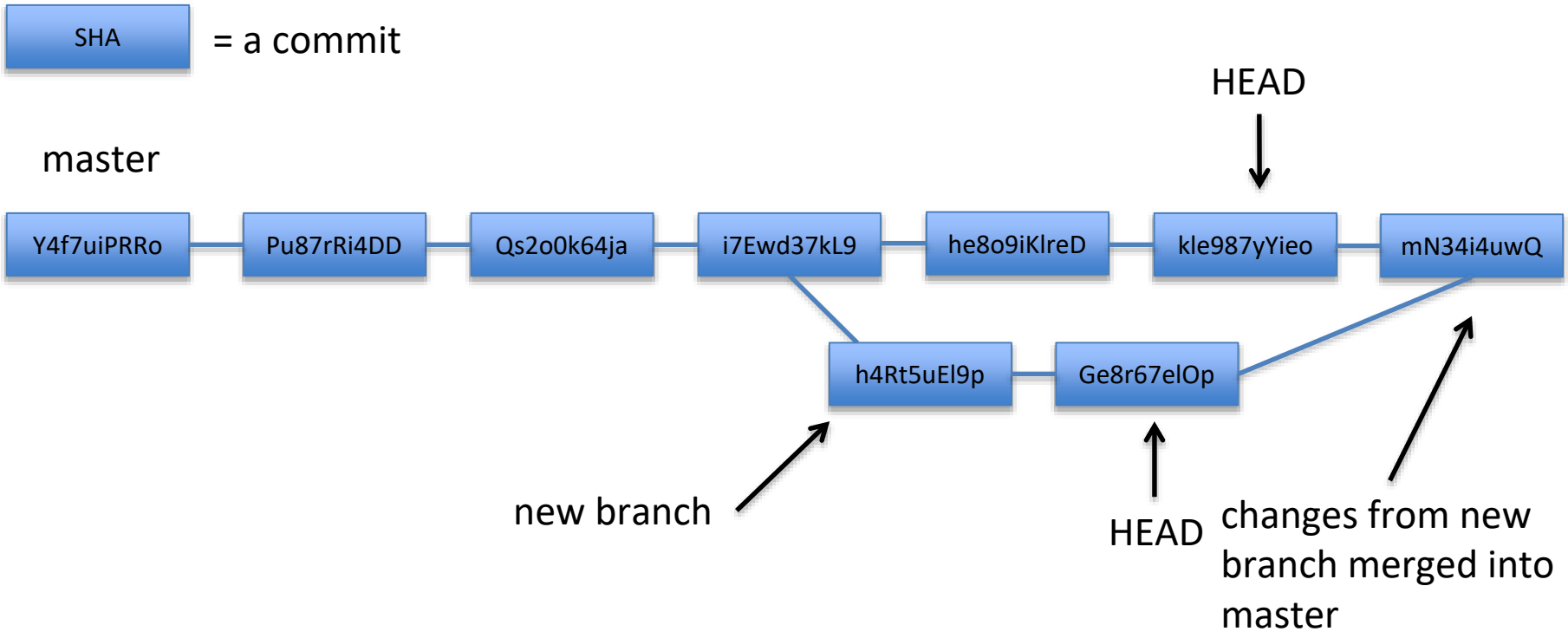
git init
git status
git log
git add
git commit
git diff
git rm
git mv

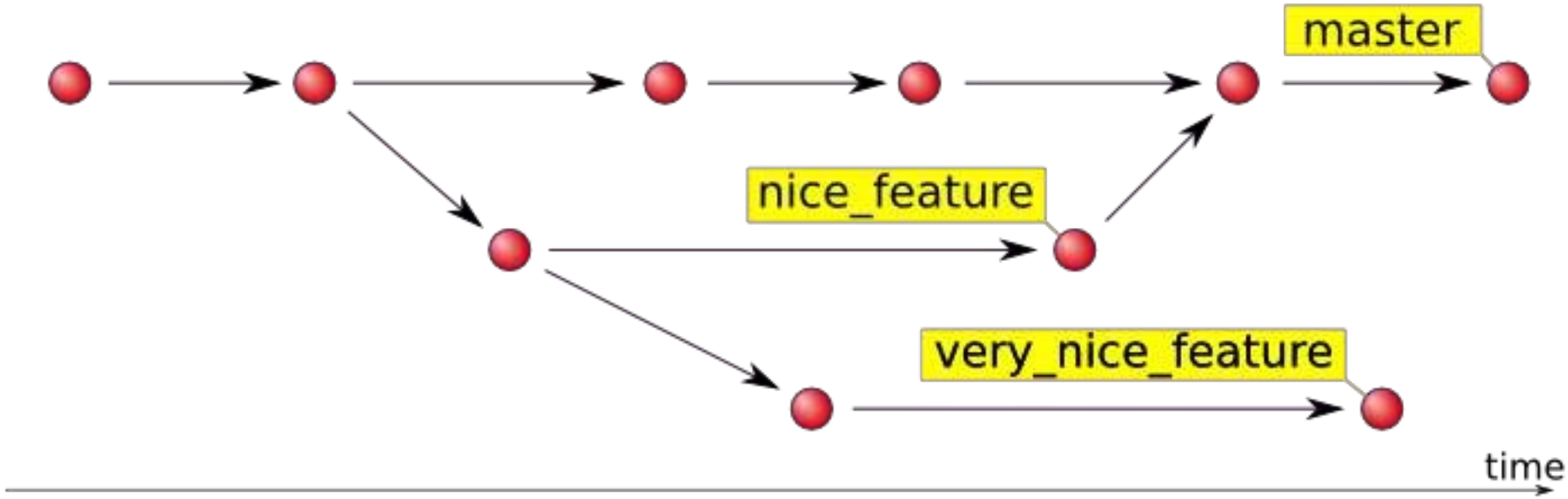


branching

- 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/

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?

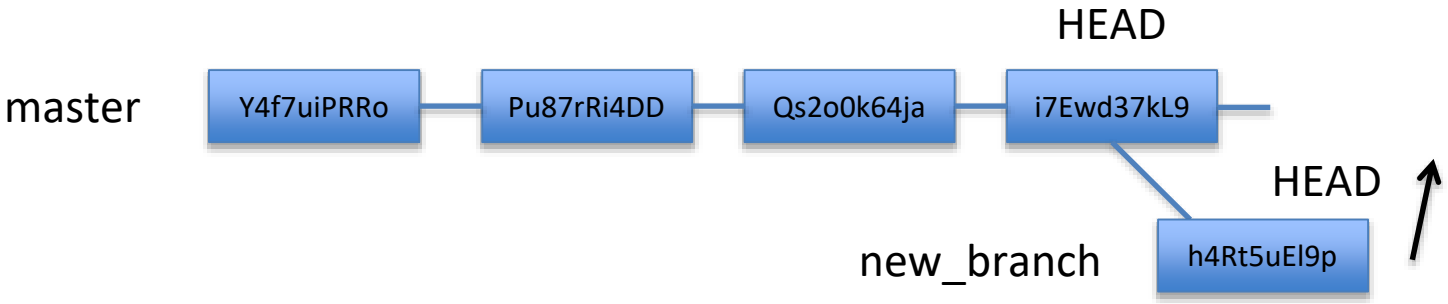
From the branch into which you want to merge another branch….

git merge branch_to_merge

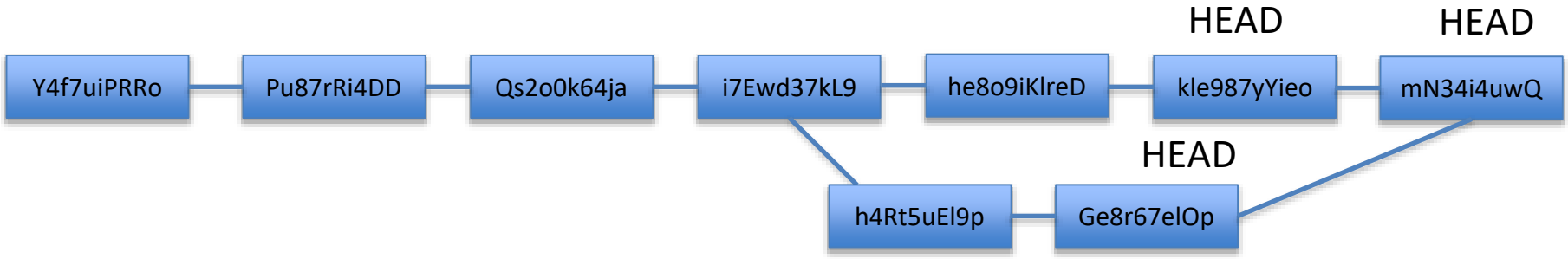
```
[dolanmi]$ git branch
* master
  new_feature
[dolanmi]$ git merge new_feature
Updating 3789cd3..1214807
Fast-forward
 file1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[dolanmi]$ git diff master..new_feature
[dolanmi]$
```

Note: Always have a clean working directory when merging

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

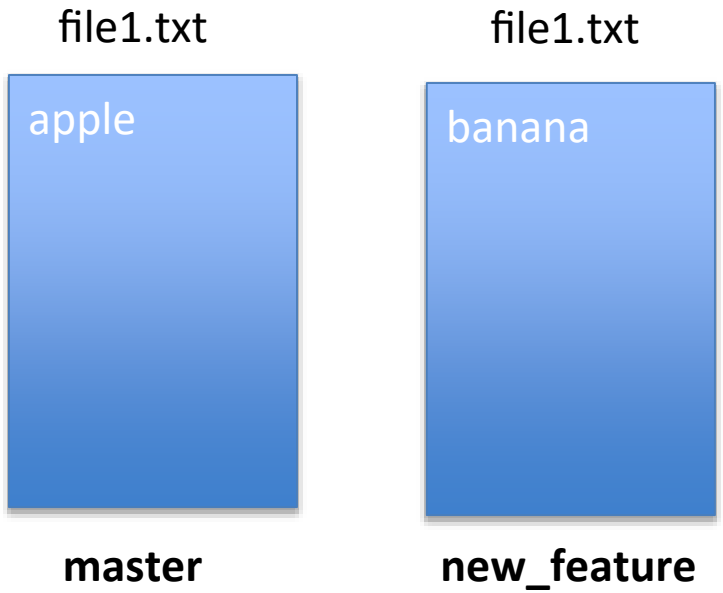


"**recursive**" merge occurs by looking back and combining ancestors to resolve merge



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

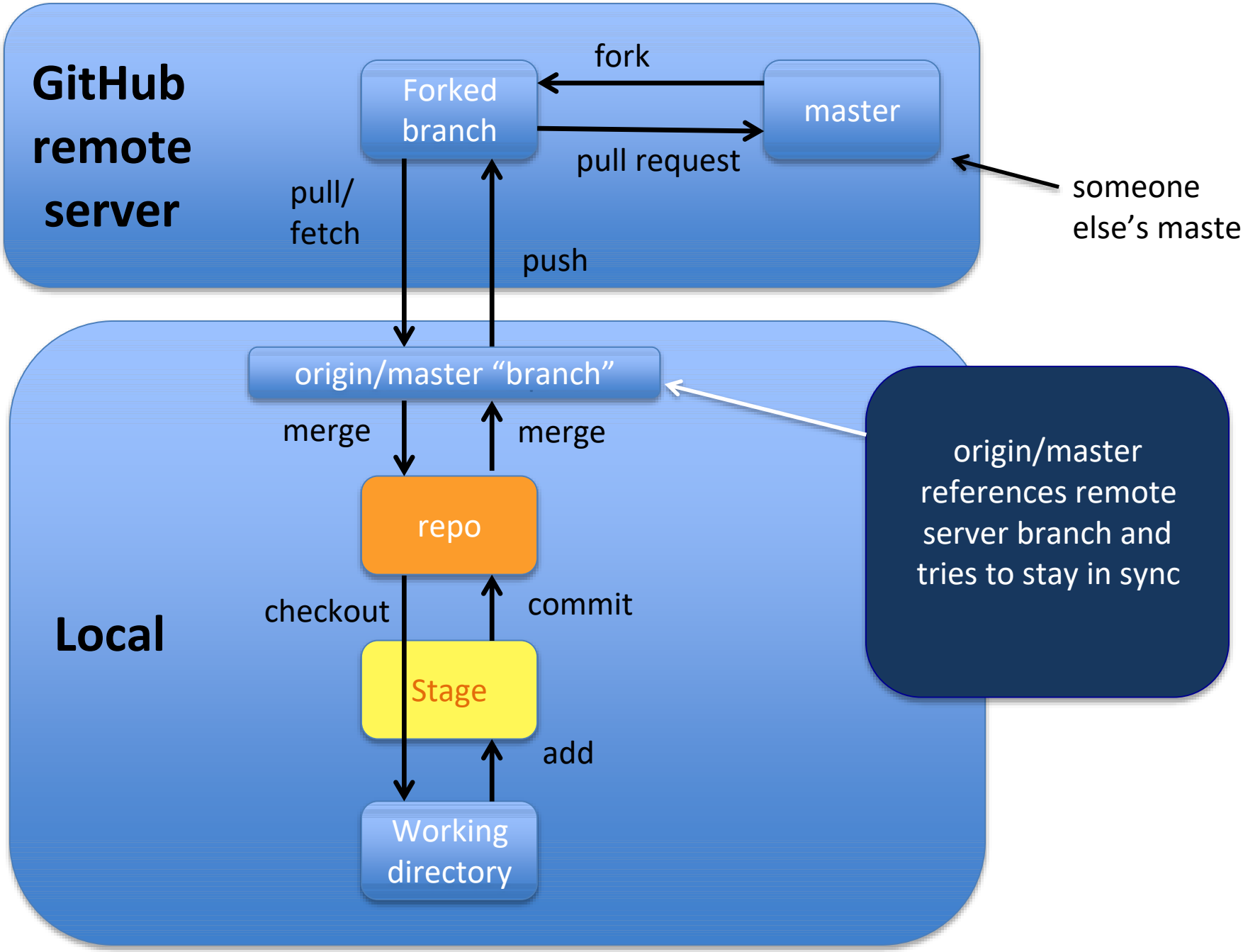
```
[dolanmi]$ git log --graph --oneline --all --decorate
*   7367e1e (HEAD -> master) fix merge conflict
|\
| * b4f09a5 (new_feature) add banana
* | df043c1 add apple
|/
* 1214807 new information added
* 3789cd3 file3.txt
* 6bfebcd new dir
* 730c6bd files
* 48f1ecf c
* 60f1c1a another message yet
*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

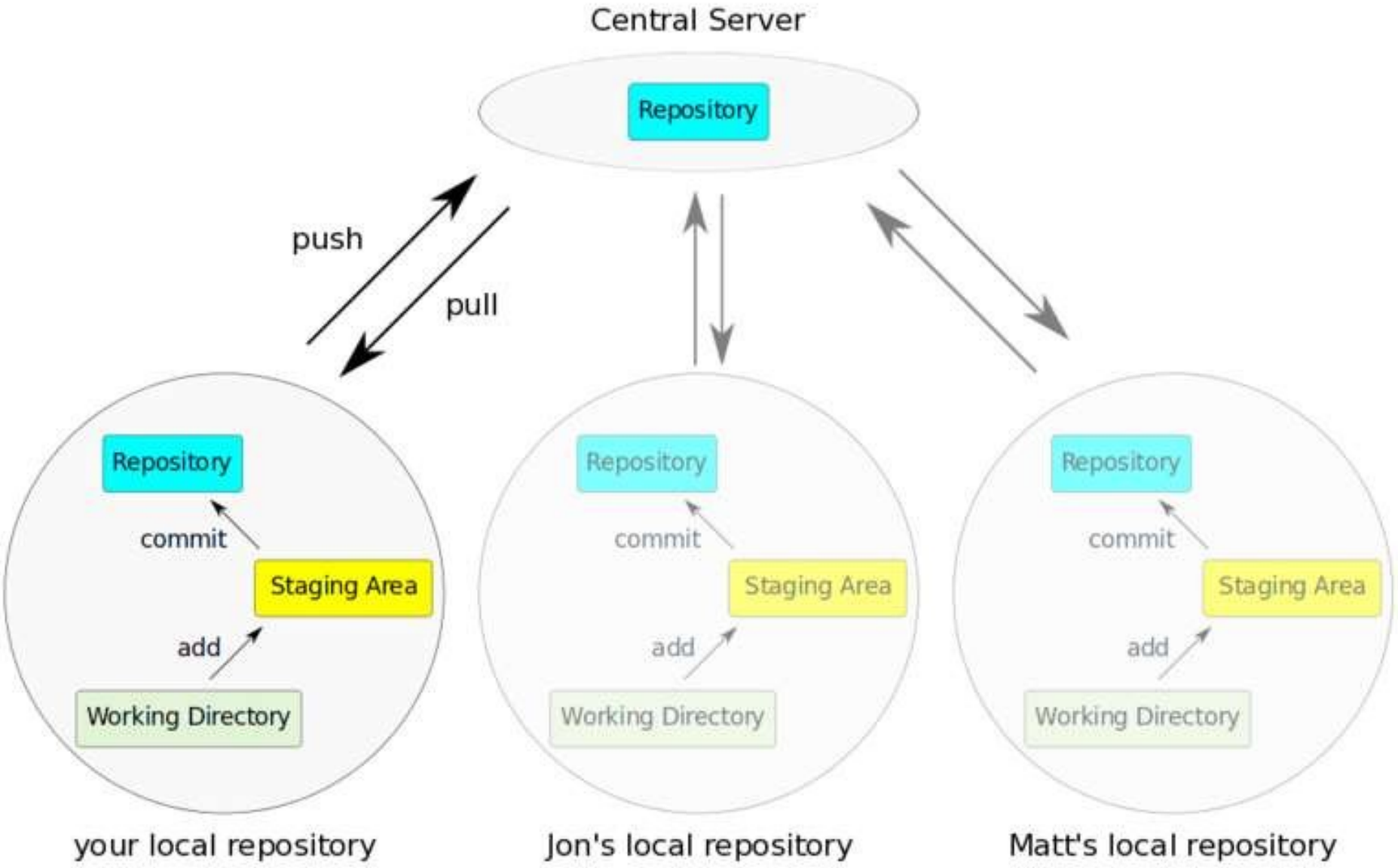
What is  ?

GitHub


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

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**

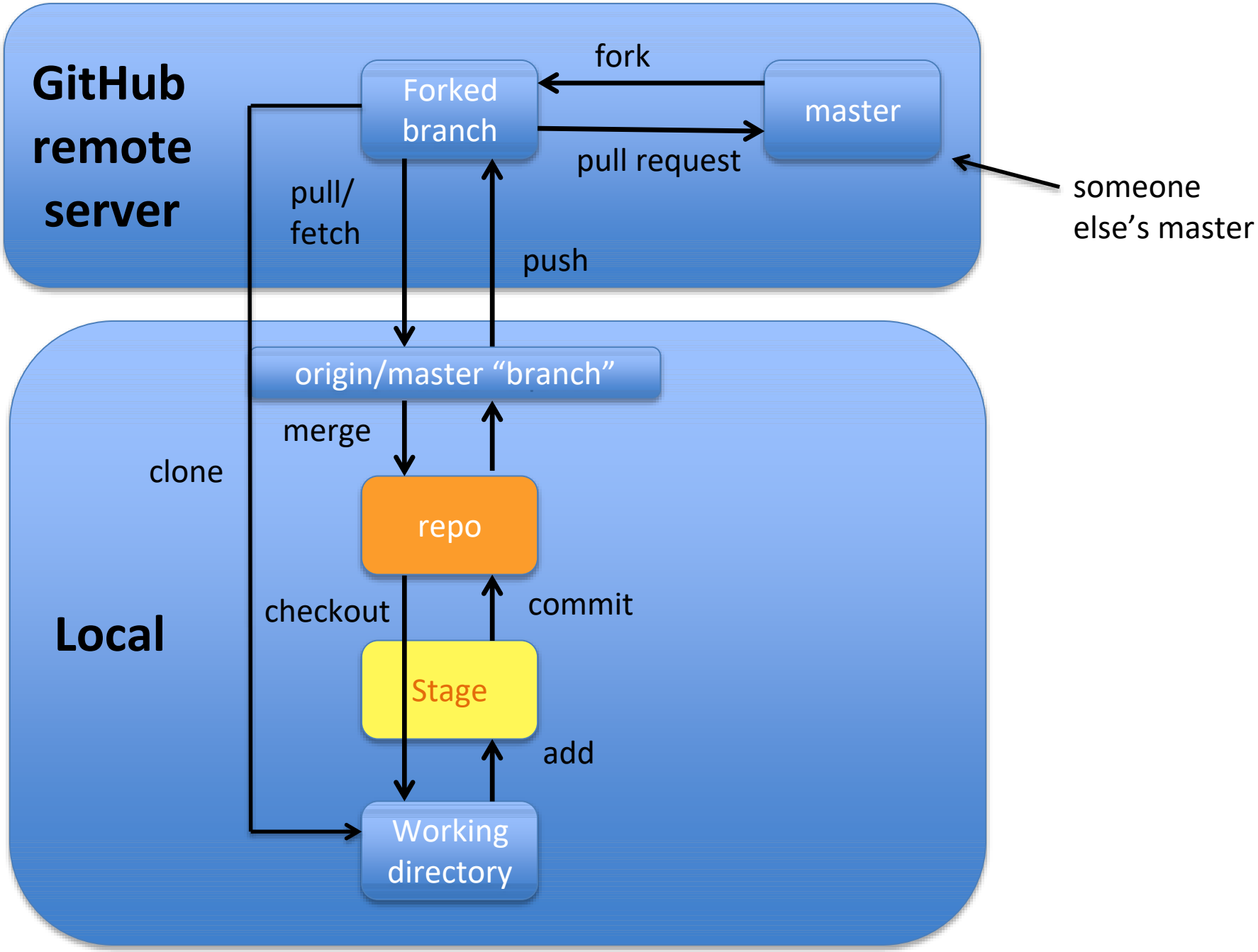


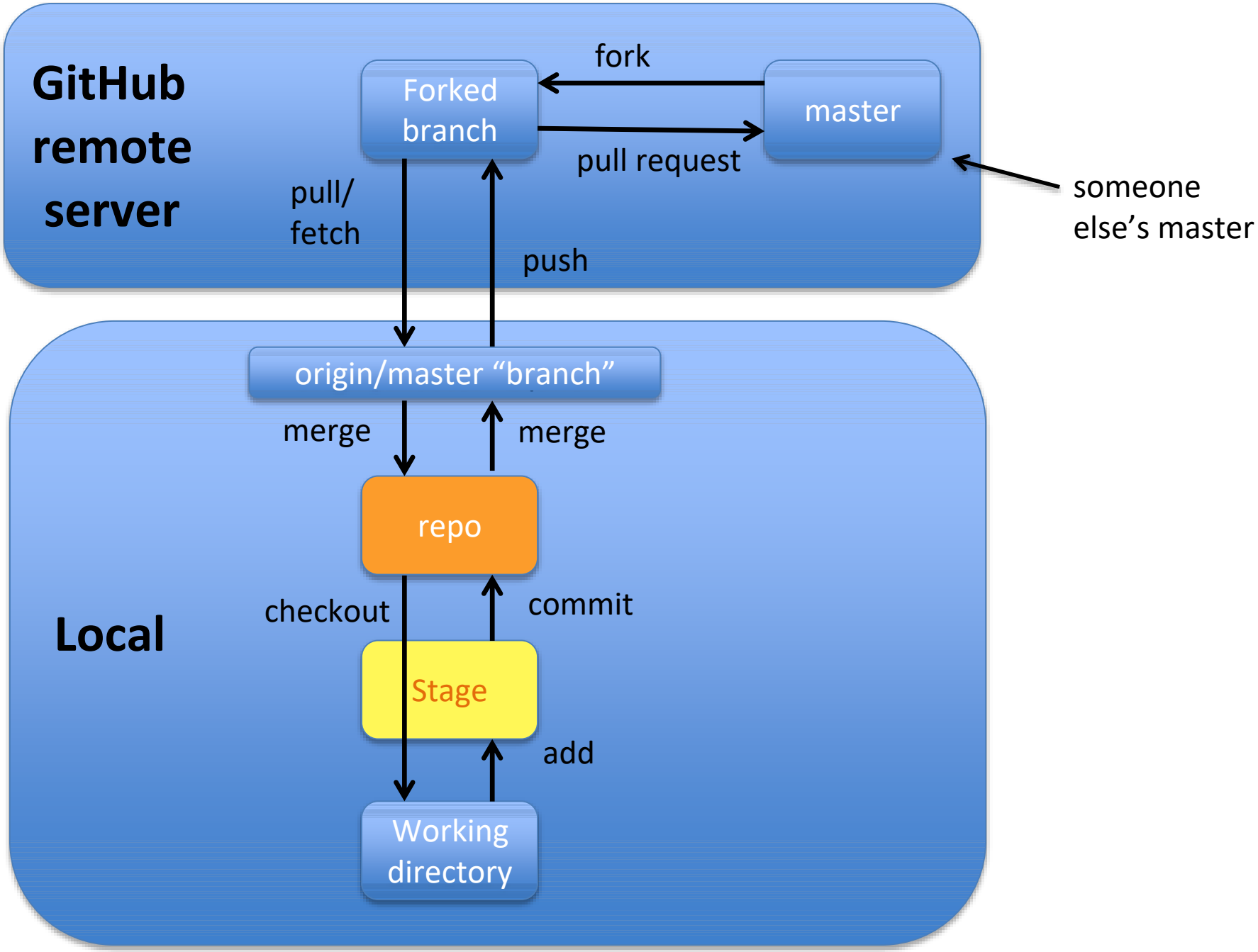
Source: http://bit.ly/1rvzjp9

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


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