CSE306 Software Quality in Practice

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
git

distributed version control system
Overview

<table>
<thead>
<tr>
<th>Last updated</th>
<th>an hour ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>—</td>
</tr>
<tr>
<td>Access level</td>
<td>Admin</td>
</tr>
<tr>
<td>Branch</td>
<td>1</td>
</tr>
<tr>
<td>Tags</td>
<td>0</td>
</tr>
<tr>
<td>Forks</td>
<td>0</td>
</tr>
<tr>
<td>Watcher</td>
<td>1</td>
</tr>
</tbody>
</table>

Invite users to this repo

Send Invitation

Recent activity

1 commit
Pushed to alphonce/classexercise
| e350833  | README.md created online with... |
| Carl Alphone · an hour ago

What is this repository for?

- Quick summary
- Version
- Learn Markdown
stash
workspace
index staging
local repository
remote repository

git clone
README

This README would normally document whatever steps are necessary to get your application up and running.

What is this repository for?

- Quick summary
- Version
- Learn Markdown

Go to #clone on this page
README

This README would normally document whatever steps are necessary to get your application up and running.

What is this repository for?

- Quick summary
- Version
- Learn Markdown
dhcp020-209-034: git alphonce$ git clone https://alphonce@bitbucket.org/alphonce/classexercise.git
git alphonce$ git clone https://alphonce@bitbucket.org/alphonce/classexercise.git
Cloning into 'classexercise'...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
dhcp020-209-034:git alphonce$ cd classeexercise/
dhcp020-209-034:classeexercise alphonce$ ls -la
dhcp020-209-034:git alphonce$ cd classexercise/
dhcp020-209-034:classexercise alphonce$ ls -la

```
total 8
drwxr-xr-x  4 alphonce  staff  136 Apr 13 09:59 .
drwxr-xr-x 29 alphonce  staff  986 Apr 13 09:59 ..
drwxr-xr-x 13 alphonce  staff  442 Apr 13 09:59 .git
-rw-r--r--  1 alphonce  staff  565 Apr 13 09:59 README.md
```
pointer to the current branch
$ more head
ref: refs/heads/master
<table>
<thead>
<tr>
<th>Mode</th>
<th>Owner</th>
<th>Group</th>
<th>Size</th>
<th>Date</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>-r</td>
<td>alphonce</td>
<td>staff</td>
<td>23</td>
<td>Apr 17</td>
<td>HEAD</td>
</tr>
<tr>
<td>drwx</td>
<td>alphonce</td>
<td>staff</td>
<td>68</td>
<td>Apr 17</td>
<td>branches</td>
</tr>
<tr>
<td>-r</td>
<td>alphonce</td>
<td>staff</td>
<td>328</td>
<td>Apr 17</td>
<td>config</td>
</tr>
<tr>
<td>-r</td>
<td>alphonce</td>
<td>staff</td>
<td>73</td>
<td>Apr 17</td>
<td>description</td>
</tr>
<tr>
<td>drwx</td>
<td>alphonce</td>
<td>staff</td>
<td>408</td>
<td>Apr 17</td>
<td>hooks</td>
</tr>
<tr>
<td>-r</td>
<td>alphonce</td>
<td>staff</td>
<td>137</td>
<td>Apr 17</td>
<td>index</td>
</tr>
<tr>
<td>drwx</td>
<td>alphonce</td>
<td>staff</td>
<td>102</td>
<td>Apr 17</td>
<td>info</td>
</tr>
<tr>
<td>drwx</td>
<td>alphonce</td>
<td>staff</td>
<td>136</td>
<td>Apr 17</td>
<td>logs</td>
</tr>
<tr>
<td>drwx</td>
<td>alphonce</td>
<td>staff</td>
<td>238</td>
<td>Apr 17</td>
<td>objects</td>
</tr>
<tr>
<td>-r</td>
<td>alphonce</td>
<td>staff</td>
<td>107</td>
<td>Apr 17</td>
<td>packed-refs</td>
</tr>
<tr>
<td>drwx</td>
<td>alphonce</td>
<td>staff</td>
<td>170</td>
<td>Apr 17</td>
<td>refs</td>
</tr>
</tbody>
</table>

staging area
$ git ls-files
README.md
content
(blobs, trees, commits)
The git man page seems to be surprisingly bereft of an official definition, other than this (emphasis mine):

The object database contains objects of three main types: blobs, which hold file data; trees, which point to blobs and other trees to build up directory hierarchies; and commits, which each reference a single tree and some number of parent commits.

The repeated use of the term "object database" across git documentation suggests a borrowing of "blob" specifically from DBMSs.

In its article on Binary large objects Wikipedia defines the term as "a collection of binary data stored as a single entity in a database management system", further offering the following:

Blobs were originally just amorphous chunks of data invented by Jim Starkey at DEC, who describes them as "the thing that ate Cincinnati, Cleveland, or whatever" from "the 1958 Steve McQueen movie", referring to The Blob.

Later, Terry McKiever, a marketing person for Apollo, felt that it needed to be an acronym and invented the backronym Basic Large Object. Then Informix invented an alternative backronym, Binary Large Object.

So, though it's not a definitive answer, the term "blob" has a conventional and well-defined usage across computer science as an opaque string of binary data, and git adheres to that definition without further specifying it.
$ ls -l objects

total 0

drwxr-xr-x  3 alphonce  staff  102 Apr 17 13:26 25

drwxr-xr-x  3 alphonce  staff  102 Apr 17 13:26 39

drwxr-xr-x  3 alphonce  staff  102 Apr 17 13:26 9c

drwxr-xr-x  2 alphonce  staff  68 Apr 17 13:26 info

drwxr-xr-x  2 alphonce  staff  68 Apr 17 13:26 pack

drwxr-xr-x  7 alphonce  staff  238 Apr 17 13:26 objects

-rw-r--r--  1 alphonce  staff  107 Apr 17 13:26 packed-refs

drwxr-xr-x  5 alphonce  staff  170 Apr 17 13:26 refs
$ git cat-file -t 25b4
commit
$ git cat-file -t 9ce9
tree
$ git cat-file -t 39af
blob
$ git cat-file -p 25b4

tree 9ce959348ab4c2eeb61549393b4b1acc0504a649
author Carl Alphonce <alphonce@buffalo.edu> 1492449992 +0000
committer Carl Alphonce <alphonce@buffalo.edu> 1492449992 +0000

README.md created online with Bitbucket
This README would normally document whatever steps are necessary to get your application up and running.

### What is this repository for? ###

* Quick summary
* Version
* [Learn Markdown](https://bitbucket.org/tutorials/markdowndemo)

### How do I get set up? ###

* Summary of set up
* Configuration
* Dependencies
* Database configuration
* How to run tests
* Deployment instructions

### Contribution guidelines ###

* Writing tests
* Code review
* Other guidelines

### Who do I talk to? ###

* Repo owner or admin
* Other community or team contact
pointers to commits

-rw-r--r-- 1 alphonce staff 23 Apr 17 13:26 HEAD
drwxr-xr-x 2 alphonce staff 68 Apr 17 13:26 branches
-rw-r--r-- 1 alphonce staff 328 Apr 17 13:26 config
-rw-r--r-- 1 alphonce staff 73 Apr 17 13:26 description
drwxr-xr-x 12 alphonce staff 408 Apr 17 13:26 hooks
-rw-r--r-- 1 alphonce staff 137 Apr 17 13:26 index
drwxr-xr-x 3 alphonce staff 102 Apr 17 13:26 info
drwxr-xr-x 4 alphonce staff 136 Apr 17 13:26 logs
drwxr-xr-x 7 alphonce staff 238 Apr 17 13:26 objects
-rw-r--r-- 1 alphonce staff 107 Apr 17 13:26 packed-refs
drwxr-xr-x 5 alphonce staff 170 Apr 17 13:26 refs
$ ls -l refs

```
total 0

drwxr-xr-x  3 alphonce staff  102 Apr 17 13:26 heads
drwxr-xr-x  3 alphonce staff  102 Apr 17 13:26 remotes
drwxr-xr-x  2 alphonce staff   68 Apr 17 13:26 tags
```

Additional directories:
- `branches`
- `config`
- `description`
- `hooks`
- `info`
- `logs`
- `objects`
- `packed-refs`
- `index`
Possible states of a file

- unmodified
- modified
- untracked
- staged

Actions:
- edit
- commit
- add
commit preserves contents  
(accidental removals can be recovered from)

edit

unmodified

commit

modified

add

staged

add

untracked
git add
create a file
add to index
(staging area)

```
git add <filename>
```
git commit
commit to local repo

```
git commit -m "message"
```
push to remote repo

- git push
pull from remote repo

- `git pull`
understanding 'reset'

Create file
git add file
git commit -m "..."
Edit file
git add file
git commit -m "..."
edit file
get add file
git commit -m "..."
Let's take this as our starting point
Moves the branch that HEAD refers to.
RESULT: master is moved; HEAD still refers to master.

```bash
git reset --soft HEAD~
```

HEAD~ is the parent of HEAD.
STARTING POINT
**git reset --mixed HEAD~**

Moves the branch that HEAD refers to, and unstages changes to the index.

RESULT: master is moved; HEAD still refers to master.

---

-- mixed is default if not specified
STARTING POINT
Moves the branch that HEAD refers to, unstages changes, and makes workspace reflect the staging area.

RESULT: master is moved; HEAD still refers to master.

```
git reset --hard HEAD~
```

-- hard can result in data loss if workspace had uncommitted files.

Moves the branch that HEAD refers to, unstages changes, and makes workspace reflect the staging area.

RESULT: master is moved; HEAD still refers to master.
git info/tutorial

- https://www.atlassian.com/git/tutorials
helpful git diagrams at this site:

http://teohm.com/blog/learning-git-internals-by-example/


more helpful git diagrams at this site (currently down?):

http://rypress.com/tutorials/git/plumbing

http://rypress.com/tutorials/git/media/12-1.png
http://rypress.com/tutorials/git/media/12-2.png