What is Git

Things you should know about Git but you've never bothered asking

Marco Biazzini

February 27, 2014

Foreword

- A stupid content tracker

Foreword

- A stupid content tracker
- Git Object Model

Foreword

- A stupid content tracker
- Git Object Model
- Take it easy!

Foreword

- A stupid content tracker
- Git Object Model
- Take it easy!

FORGET ABOUT SVN !!!

Foreword

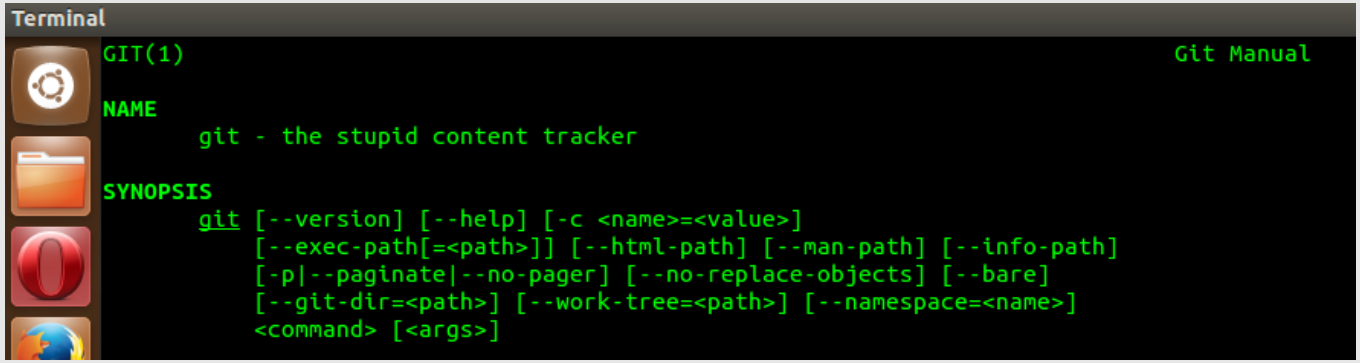
- A stupid content tracker
- GIT Object Model
- Take it easy!

FORGET ABOUT SVN !!!

Reference : Scott Chacon's PROGIT.

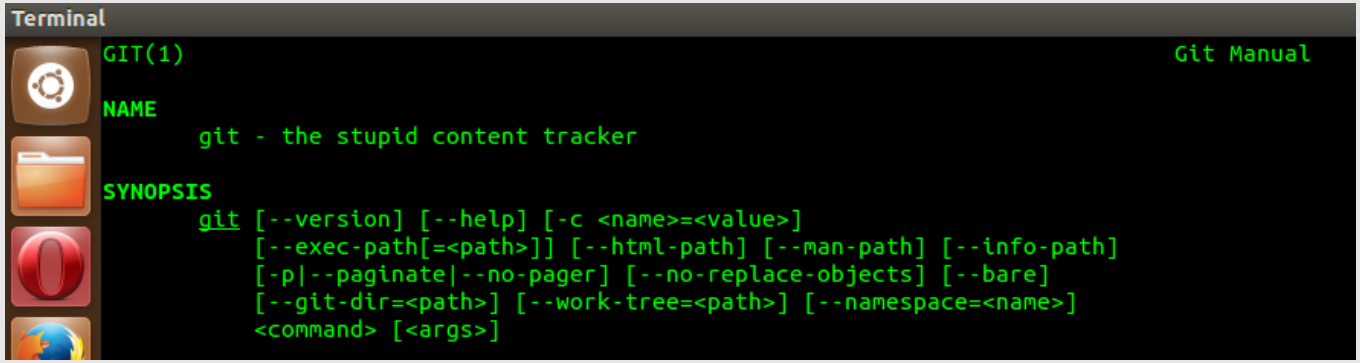
Outline

A stupid content tracker



What Mr. Torvalds thinks of his creature...

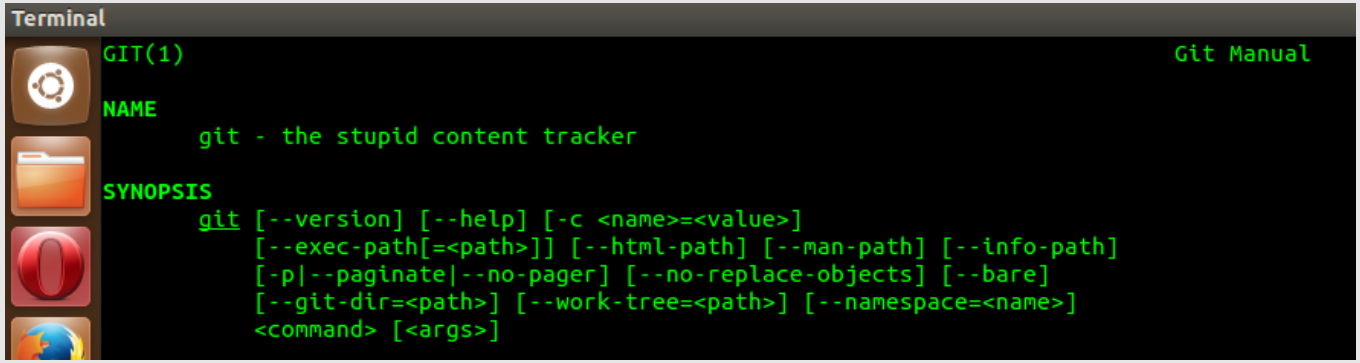
A stupid content tracker



What Mr. Torvalds thinks of his creature...

How does one build a content tracker?

A stupid content tracker



What Mr. Torvalds thinks of his creature...

How does one build a content tracker?
One builds a system based on two simple ideas:

- hashing a content to uniquely identify it.
- being able to retrieved any hashed content.

A stupid content tracker

The whole GIT system is built on these ideas:

- hash a content :
  `git hash-object <content>`
- retrieve a hashed content :
  `git cat-file <op> <hash>`

A stupid content tracker

The whole GIT system is built on these ideas:

- hash a content :
  `git hash-object <content>`
- retrieve a hashed content :
  `git cat-file <op> <hash>`

... And, by the way: what is a content?

A stupid content tracker

The whole GIT system is built on these ideas:

- hash a content :
  `git hash-object <content>`
- retrieve a hashed content :
  `git cat-file <op> <hash>`

... And, by the way: what is a content?
→ *Anything* that I want to be tracked
whenever I want it to be tracked!

Hacking up a GIT repo

1. $ git init
   Initialized empty Git repository in <‘pwd‘>/.git/

Hacking up a GIT repo

1. $ git init
   Initialized empty Git repository in <‘pwd‘>/.git/
2. $ echo "ciao" > Marco.txt
3. $ git hash-object -w Marco.txt
   887ae9333d92a1d72400c210546e28baa1050e44

Hacking up a GIT repo

1. $ git init
   Initialized empty Git repository in <‘pwd‘>/.git/
2. $ echo "ciao" > Marco.txt
3. $ git hash-object -w Marco.txt
   887ae9333d92a1d72400c210546e28baa1050e44
4. $ echo "ciao" > Mario.txt
5. $ git hash-object -w Mario.txt
   887ae9333d92a1d72400c210546e28baa1050e44

Hacking up a GIT repo

1. $ git init
   Initialized empty Git repository in <‘pwd‘>/.git/
2. $ echo "ciao" > Marco.txt
3. $ git hash-object -w Marco.txt
   887ae9333d92a1d72400c210546e28baa1050e44
4. $ echo "ciao" > Mario.txt
5. $ git hash-object -w Mario.txt
   887ae9333d92a1d72400c210546e28baa1050e44
6. $ find .git/objects/ -type f
   .git/objects/88/7ae9333d92a1d72400c210546e28baa1050e44
7. $ git cat-file -p 887ae
   ciao

Hacking up a GIT repo

8. $ echo "ciao anche a te" >> Marco.txt

Hacking up a GIT repo

8. $ echo "ciao anche a te" >> Marco.txt
9. $ git update-index --add Marco.txt
10. $ git update-index --add Mario.txt
11. $ git write-tree
    10fe3ecb8aa002edc21faea0a6671a4ce1e27b79

Hacking up a GIT repo

8. $ echo "ciao anche a te" >> Marco.txt
9. $ git update-index --add Marco.txt
10. $ git update-index --add Mario.txt
11. $ git write-tree
    10fe3ecb8aa002edc21faea0a6671a4ce1e27b79
12. $ find .git/objects/ -type f
    .git/objects/cc/b7a476fec518a19793740d1b7dc0b806b32ff4
    .git/objects/88/7ae9333d92a1d72400c210546e28baa1050e44
    .git/objects/10/fe3ecb8aa002edc21faea0a6671a4ce1e27b79

Hacking up a GIT repo

8. $ echo "ciao anche a te" >> Marco.txt
9. $ git update-index --add Marco.txt
10. $ git update-index --add Mario.txt
11. $ git write-tree
    10fe3ecb8aa002edc21faea0a6671a4ce1e27b79
12. $ find .git/objects/ -type f
    .git/objects/cc/b7a476fec518a19793740d1b7dc0b806b32ff4
    .git/objects/88/7ae9333d92a1d72400c210546e28baa1050e44
    .git/objects/10/fe3ecb8aa002edc21faea0a6671a4ce1e27b79
13. $ git cat-file -p 10fe3e

    100644 blob ccb7a476fec518a19793740d1b7dc0b806b32ff4 Marco.txt

    100644 blob 887ae9333d92a1d72400c210546e28baa1050e44 Mario.txt

Hacking up a GIT repo

14. $ echo 'first commit' | git commit-tree 10fe3e
    bb9ba5233134a8c2fcf738a524e0dd6baec54811

Hacking up a Git repo

14. $ echo 'first commit' | git commit-tree 10fe3e
    bb9ba5233134a8c2fcf738a524e0dd6baec54811
15. $ find .git/objects/ -type f
    .git/objects/cc/b7a476fec518a19793740d1b7dc0b806b32ff4
    .git/objects/bb/9ba5233134a8c2fcf738a524e0dd6baec54811
    .git/objects/88/7ae9333d92a1d72400c210546e28baa1050e44
    .git/objects/10/fe3ecb8aa002edc21faea0a6671a4ce1e27b79

Hacking up a GIT repo

14. $ echo 'first commit' | git commit-tree 10fe3e
    bb9ba5233134a8c2fcf738a524e0dd6baec54811
15. $ find .git/objects/ -type f
    .git/objects/cc/b7a476fec518a19793740d1b7dc0b806b32ff4
    .git/objects/bb/9ba5233134a8c2fcf738a524e0dd6baec54811
    .git/objects/88/7ae9333d92a1d72400c210546e28baa1050e44
    .git/objects/10/fe3ecb8aa002edc21faea0a6671a4ce1e27b79
16. $ echo "ciao" > Marco.txt
17. $ echo "eppur si muove" > Galileo.txt
18. $ git update-index --add Galileo.txt
19. $ git update-index Marco.txt
20. $ git write-tree
    ff4b1a22e204a58c91df73b05b001e9833c1ea10

Hacking up a GIT repo

21. $ echo 'second commit' | git commit-tree ff4b1 -p bb9ba5
    4a8857b78d7a5847e555ca35b00b6ee7b1c7e27f
22. $ git log --format=raw 4a885

```
commit 4a8857b78d7a5847e555ca35b00b6ee7b1c7e27f
tree ff4b1a22e204a58c91df73b05b001e9833c1ea10
parent bb9ba5233134a8c2fcf738a524e0dd6baec54811
author Marco Biazzini <Marco.Biazzini@inria.fr> 1393484216 +0100
committer Marco Biazzini <Marco.Biazzini@inria.fr> 1393484216 +0100

    second commit

commit bb9ba5233134a8c2fcf738a524e0dd6baec54811
tree 10fe3ecb8aa002edc21faea0a6671a4ce1e27b79
author Marco Biazzini <Marco.Biazzini@inria.fr> 1393483888 +0100
committer Marco Biazzini <Marco.Biazzini@inria.fr> 1393483888 +0100

    first commit
```

Hacking up a GIT repo

References : humanly named files which point to an object hash.
They can be found in

- `.git/refs/heads` if they are branches heads;
- `.git/refs/tags` if they are simple tags;
- `.git/refs/remotes` if they are remote heads.

Hacking up a GIT repo

References : humanly named files which point to an object hash.
They can be found in

- `.git/refs/heads` if they are branches heads;
- `.git/refs/tags` if they are simple tags;
- `.git/refs/remotes` if they are remote heads.

ANY GIT object can be pointed by a ref or a tag!
Even a ref itself, by a symbolic ref (e.g. : HEAD).

Hacking up a GIT repo

References : humanly named files which point to an object hash.
They can be found in

- `.git/refs/heads` if they are branches heads;
- `.git/refs/tags` if they are simple tags;
- `.git/refs/remotes` if they are remote heads.

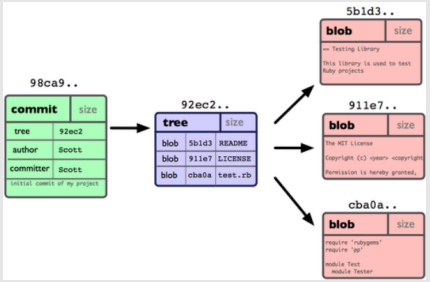
ANY GIT object can be pointed by a ref or a tag!
Even a ref itself, by a symbolic ref (e.g. : HEAD).

To complete the hack :

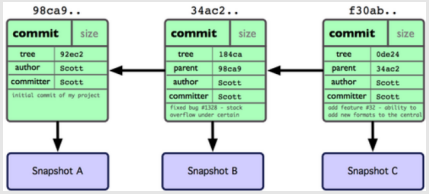
```
echo "4a8857b78d7a5847e555ca35b00b6ee7b1c7e27f" > .git/refs/heads/master
```

The Git Object Model

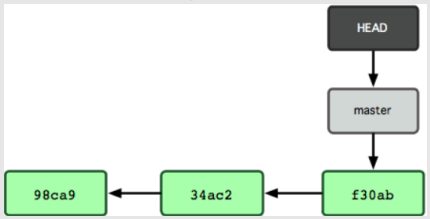
A basic Git repo contains these objects:



A commit chain in the history looks like this:



And so a commit graph looks like this:



The Git Object Model

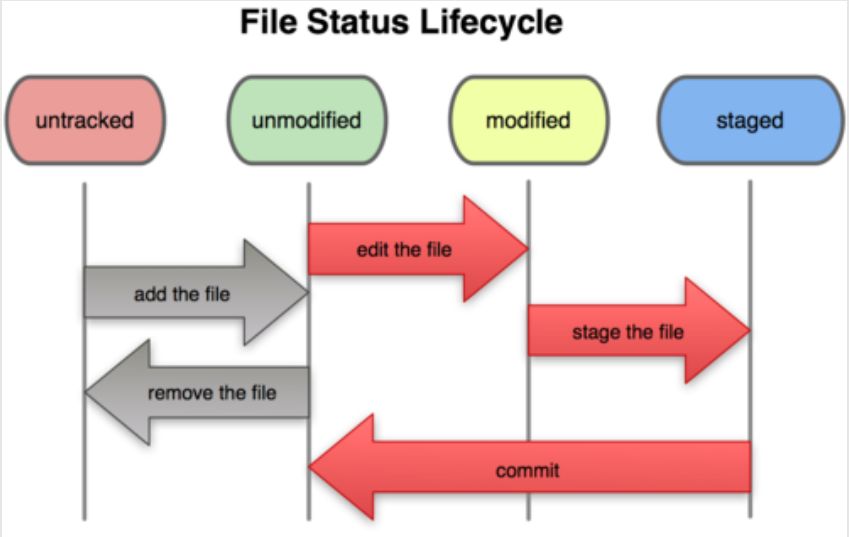
How does a Git repo look like from inside?

The Git Object Model

How does a Git repo look like from inside?

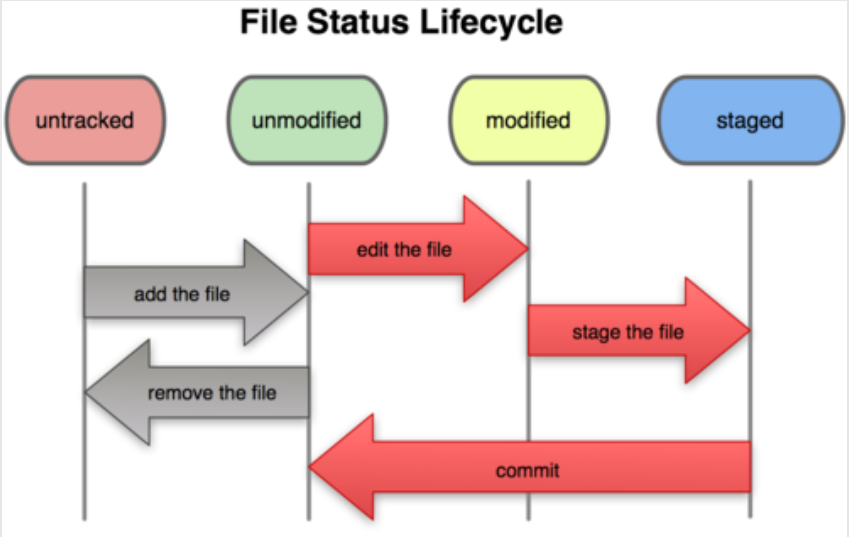
```
$ ls -1 .git/
  branches/      # no longer used
  info/          # global exclude file
  description    # used by GitWeb only
  index          # tracked files
  hooks/         # automation scripts
  objects/       # all contents are here
  refs/          # branches and tags
  HEAD           # the current branch HEAD
  config         # project–specific configuration
```

Gitters do it better



- git status
- git add
- git commit
- git rm
- git diff

Gitters do it better



- git status
- git add
- git commit
- git rm
- git diff

Other cool stuff:

- recover lost commits : `git log -g`
- delete things from history:
  `git filter-branch + git prune`

Gitters do it better

Now some things may be a little less misterious. . .

- *git clone* = retrieve the '.git/' and checkout from HEAD;
- *git branch* = add a one–line file in .git/refs/heads/
- *git merge* = if no conflict, add objects in .git/objects/ and a ref in ./git/refs/heads/
- *git remote* = add some lines into .git/config
- *git fetch* = add a ref in .git/refs/remotes and some objects in .git/objects/
- *git pull* = git fetch + git merge

Gitters do it better

Final suggestions :

- `man git` is your friend, do not neglect it!
- ProGit manual: chapters 1-4 are highly suggested.
- Do not be afraid to check up on the internals (.git/config, .git/refs/, ...)
- Nothing is lost, if it has been committed!
- Train! It's free! Check out:
  `http://pcottle.github.io/learnGitBranching/?demo`