A TALE OF THREE TREES

a magical afternoon with Scott Chacon
About Me
Git Resources

git-scm.com

gitref.org

progit.org
@chacon
<\/me>
PREFACE
How Git Works
It's all about the trees, baby
TREE IS
files and subtrees
EXAMPLE
$ tree

.  ├── README
    │   └── README
    └── Rakefile
        └── lib
            └── git.rb

1 directory, 3 files
ACT ONE
The Three Trees
first tree
the HEAD
$ cat .git/HEAD
ref: refs/heads/master

$ cat .git/refs/heads/master
e9a570524b63d2a2b3a7c3325acf5b89bbeb131e

$ git cat-file -p e9a570524b63d2a2b3a7c3325acf5b89bbeb131e
tree cfda3bf379e4f8dba8717dee55aab78aef7f4daf
author Scott Chacon <schacon@gmail.com> 130151183
committer Scott Chacon <schacon@gmail.com> 130151

initial commit

$ git ls-tree -r cfda3bf379e4f8dba8717dee55aab78aef7f4daf
tree 99f1a6d12cb4b6f19... lib
dir 99f1a6d12cb4b6f19... lib
100644 blob a906cb2a4a904a152... README
100644 blob 8f94139338f9404f2... Rakefile
second tree
the index
The Staging Area
require 'rugged'

index = Rugged::Index.new("/opt/repo/.git/index")
index.refresh

index.each do |entry|
  puts "File Name: " + entry.path
  puts " Blob SHA: " + entry.sha
  puts "File Size: " + entry.file_size.to_s
  puts "File Mode: " + entry.mode.to_s
  puts " mtime: " + entry.mtime.to_i.to_s
  puts " ctime: " + entry.ctime.to_i.to_s
  puts " Inode: " + entry.ino.to_s
  puts " UID: " + entry.uid.to_s
  puts " GID: " + entry.gid.to_s
end
third tree

the working directory
$ tree

.  
  ├── .git
  │   ├── HEAD
  │   └── index
  ├── README
  └── lib
      └── git.rb
Three Trees
HEAD, Index and Working Directory
Checkout the project

Stage files

Commit

HEAD

Index

Working Directory
Tree Roles

**HEAD** last commit, next parent

**Index** proposed next commit

**Work Dir** sandbox
ACT TWO
Working With Trees
git status
$ git status
# On branch master
# Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
# Changes to be committed:
#  (use "git reset HEAD ..." to unstage)
#   modified:   jobs/email_reply.rb
#
# Changed but not updated:
#  (use "git add ..." to update what will be committed)
#  (use "git checkout -- ..." to discard changes in working directory)
#   modified:   app/helpers/users_helper.rb
#   modified:   test/unit/email_reply_job_test.rb
$ git status
# On branch master
# Your branch is behind 'origin/master' by 2 commits,
# and can be fast-forwarded.
#
# Changes to be committed:
#   HEAD and index differ
#
#   modified:   jobs/email_reply.rb
#
# Changed but not updated:
#   index and working directory differ
#
#   modified:   app/helpers/users_helper.rb
#   modified:   test/unit/email_reply_job_test.rb
Git Repository

HEAD

master

?}

git add

HEAD

Index

v1 file.txt

Working Directory

v1 file.txt
$ git status

# On branch master
# Your branch is behind 'origin/master' by 2 commits,
# and can be fast-forwarded.
#
# Changed but not updated:
#  (use "git add ..." to update working directory)
#  (use "git checkout -- ..." to d
#    in working directory)
#
  modified:   file.txt
Git Repository

HEAD
---
master
---
eb43bf8
---
v1 file.txt

HEAD
---
eb43bf8
---
v1 file.txt

Index
---
v1 file.txt

Working Directory
---
v2 file.txt

edit file
Git Repository

```
HEAD
master
```

```
eb43bf8
```

```
v1   file.txt
```

```
Working Directory
```

```
HEAD
```

```
eb43bf8
```

```
v1   file.txt
```

```
Index
```

```
v2   file.txt
```

```
Working Directory
```

```
v2   file.txt
```

```git add```
$ git status
# On branch master
# Your branch is behind 'origin/master' and can be fast-forwarded.
#
# Changes to be committed:
# (use "git reset HEAD ..." to un
#
#   modified: file.txt
#
#
git add
git commit
git reset
2 forms
git reset [commit] [path]

git reset [commit]
1. Path Form

git reset [commit] [path]
git reset [file]
is the opposite of
git add [file]
git reset file.txt
git add file.txt
Reset to an older file
git reset eb43 -- file.txt
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD ..." to un
#   
#     modified:   file.txt
#
# Changed but not updated:
#   (use "git add ..." to update what
#   (use "git checkout -- ..." to d
#   
#     modified:   file.txt
#   

2. Commit Form

`git reset [commit]`
Reset Options

--soft  move HEAD to target

[--mixed] then copy to index

--hard  then copy to work dir
--soft
move HEAD to another commit
git reset --soft HEAD~
--mixed
move HEAD to another commit, then copy into index
git reset [--mixed] HEAD~
--hard
move HEAD, copy to index, copy to working directory
git reset --hard HEAD
whythefuckwouldieverwanttousethis
unstaging changes
git reset
OR

git reset [file]
- Git reset command

Diagram showing:
- HEAD
- master
- Use of commit hash eb43bf8
- v1 file.txt
- Git Repository

Working Directory:
- v2 file.txt

Index:
- v1 file.txt

HEAD:
- eb43bf8
- v1 file.txt
undo last commit
git reset [--mixed] HEAD~

moves HEAD back and moves index back
git reset [--mixed] HEAD~
undo last commit

but keep the stage
git reset --soft HEAD~

moves HEAD back but keeps index
git reset --soft HEAD~
git reset [--mixed] HEAD~
squash the last 2 commits into one
git reset --soft HEAD~2

git commit

moves HEAD back, keeps index
git reset --soft HEAD~2
Certificate of Achievement

This to certify that

you people

has successfully completed

Git Reset Training

Almost DC Apr 1, 11

Location Date

Scott “Dragon” Chacon

Course Manager
just a bit outside

tried the corner and missed
2 forms
git checkout [commit] [path]

git checkout [commit]
# Reset v. Checkout

[resetvcheckout](https://schacon.github.com/resetvcheckout.html)

<table>
<thead>
<tr>
<th>Commit Level</th>
<th>HEAD</th>
<th>Index</th>
<th>Work Dir</th>
<th>WD Safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>reset --soft [commit]</td>
<td>REF</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>reset [commit]</td>
<td>REF</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>reset --hard [commit]</td>
<td>REF</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>checkout [commit]</td>
<td>HEAD</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File Level</th>
<th>HEAD</th>
<th>Index</th>
<th>Work Dir</th>
<th>WD Safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>reset (commit) [file]</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>checkout (commit) [file]</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>
ACT THREE
Fun With Your Trees
Patchy Work

git add --patch [file]
git reset --patch (commit) [file]
git checkout --patch (commit) [file]
git read-tree
git write-tree
$ ls
README     Rakefile   lib

$ git init
Initialized empty Git repository in /private/tmp/gt/.git/
nothing added to commit but untracked files present

$ git log
fatal: bad default revision 'HEAD'

$ git add --all

$ git write-tree
4b8ad0172510761cb0e07d2c4220932bf41bbd07

$ git ls-tree 4b8ad0172510761cb0e07d2c4220932bf41bbd07
100644 blob 45dc653de6860... README
100644 blob ea3fe2ac46e92... Rakefile
040000 tree 99f1a6d12cb4b... lib
git commit-tree
Environment Variables
moving around your trees
GIT_DIR
$ mv .git /opt/repo.git
$ git --git-dir=/opt/repo.git log
$ export GIT_DIR=/opt/repo.git
$ git log
GIT_INDEX_FILE
$ git status -s
  M README
  M kidgloves.rb

$ git add kidgloves.rb

$ git status -s
  M README
  M README
  S kidgloves.rb
$ export GIT_INDEX_FILE=/tmp/index
$ git read-tree HEAD
$ git add README

$ git status -s
S   README
   M kidgloves.rb
$ unset GIT_INDEX_FILE

$ git status -s
  M README
S  kidgloves.rb

$ export GIT_INDEX_FILE=/tmp/index

$ git status -s
  S  README
S  README
  M kidgloves.rb
GIT_WORK_TREE
$ git status -s
  M README
S  kidgloves.rb

$ export GIT_DIR=$(pwd)/.git
$ export GIT_WORK_TREE=$(pwd)

$ cd /tmp
$ git status -s
  M README
S  kidgloves.rb
WTFIEWTUT

whythefuckwouldieverwanttousethis
Publishing Docs to Another Branch

task :publish_docs do
  `rocco libgit.rb` # creates libgit.html
  ENV['GIT_INDEX_FILE'] = '/tmp/i'
  `git add -f libgit.html`
  tsha = `git write-tree`
  csha = `echo 'boom' | git commit-tree #{tsha}`
  `git update-ref refs/heads/gh-pages #{csha}`
  `git push -f origin gh-pages`
end
Making Tarballs of Project Subsets
rm /tmp/in
ENV['GIT_INDEX_FILE'] = '/tmp/in'
git read-tree --prefix lib master:lib
`git read-tree --prefix ext-m extras:ext`
tsha = `git write-tree`
git archive --format=zip -o out.zip #{tsha}`
$ unzip out.zip | head
Archive:  out.zip
    creating: ext-m/
    creating: ext-m/java/
    creating: ext-m/java/nokogiri/
inflating: ext-m/java/nokogiri/EncodingHandler.java
inflating: ext-m/java/nokogiri/HtmlDocument.java
inflating: ext-m/java/nokogiri/HtmlElementDescription.java
inflating: ext-m/java/nokogiri/HtmlEntityLookup.java
inflating: ext-m/java/nokogiri/HtmlSaxParserContext.java
inflating: ext-m/java/nokogiri/NokogiriService.java
back_branch = 'refs/heads/backup'

`rm /tmp/backup_index`

ENV['GIT_INDEX_FILE'] = '/tmp/backup_index'

last_commit = `git rev-parse #{back_branch}`.strip
last_tree = `git rev-parse #{back_branch}^{{tree}}`.strip

`git add --all`
next_tree = `git write-tree`.strip

if last_tree != next_tree
    extra = last_commit.size == 40 ? "-p #{last_commit}" : ''
    csha = `echo 'back' | git commit-tree #{next_tree} #{extra}`
    `git update-ref #{back_branch} #{csha}`
end
$ git log backup
commit e3219f9d18ac485f563995a39c139736abd75420
Author: Scott Chacon <schacon@gmail.com>
Date: Thu Mar 31 13:51:05 2011 -0700

back

commit cff888a65f56572358bdd233fe6af46c48f1d36d
Author: Scott Chacon <schacon@gmail.com>
Date: Thu Mar 31 13:50:54 2011 -0700

back

commit 15f3b561b351187bf712037f267036b90438c987
Author: Scott Chacon <schacon@gmail.com>
Date: Thu Mar 31 13:45:56 2011 -0700

back
Freeze Submodules
Before Push
current_commit = `git rev-parse HEAD`
current_tree = `git rev-parse HEAD^\{tree}\`

# get a list of submodules
status = `git submodule status`.chomp
subdata = status.split("\n")
subdata.each do |subline|
  sharaw, path = subline.split(" ")
  sha = sharaw[1, sharaw.size - 1]
  remote = path.gsub('/', '-')
  `git remote add #{remote} #{path} 2>/dev/null` # fetch each submodule
  `git fetch #{remote}`
  `git read-tree --prefix=#{path} #{sha}` # for each submodule/sha end

# find heroku parent
prev_commit = `git rev-parse heroku 2>/dev/null`.chomp
pcommit = (prev_commit != "heroku") ? "-p #{prev_commit}" : ''

# write-tree/commit-tree with message of what commit sha it's based on
tree_sha = `git write-tree`.chomp
commit_sha = `echo "deploy at #{current_commit}" | git commit-tree

# update-ref
`git update-ref refs/heads/heroku #{commit_sha}`

# reset
`git reset HEAD`
CODA
Let's Review
Tree Roles

**HEAD** last commit, next parent

**Index** proposed next commit

**Work Dir** sandbox
Hi, you've reached Jimmy

If you can dream it, you can do it!
thanks!
questions?
questions?

threetrees.herokuapp.com
schacon.github.com/resetvcheckout.html
github.com/schacon/tale_of_three_trees
gist.github.com/582888