Chapter 1

[illegible]

[illegible]GIT

[illegible]Git, [illegible]

[illegible]Git[illegible]

[illegible]Git[illegible]20[illegible]

[illegible]Git[illegible]90%[illegible]30[illegible]

[illegible]**Git**[illegible], [illegible]

[illegible]**Git**[illegible],[illegible]Git[illegible] [illegible]Git[illegible]

[illegible]Git, [illegible]**Git**[illegible] Git [illegible]Git [illegible]

[illegible]**low-level documentation**[illegible]Git hacker [illegible]Git [illegible]

[illegible]

[illegible]emailschacon@gmail.com, [illegible]git[illegible] [illegible](source)http://github.com/schacon/gitbook, [illegible](patch)[illegible]pull[illegible]

[illegible]:[illegible],[illegible]emalluhui998@gmail.com, [illegible]git[illegible] [illegible](source)http://github.com/liuhui998/gitbook, [illegible](patch)[illegible]pull[illegible]

[illegible]

[illegible]url:

- Git User Manual
- The Git Tutorial
- The Git Tutorial pt 2
- "My Git Workflow" blog post

GIT□□□□

SHA

□□□□□□□□□□□□□□□,□□□□□40□□□□□40-digit□“□□□”□□□□□□□□□□□□□□:

```
6ff87c4664981e4397625791c8ea3bbb5f2279a3
```

□□□Git□□□□□□□“40□□□”□□□□□□□“□□□”□□□“□□”□□□SHA1□□□□□□□□□SHA1□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□“□□□”□

□□□□□□□□□□

- Git□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□repository□□“□□□”□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□“□□□”□□
- Git□□□□□□□□□□□□SHA1□□□□□“□□□”□□□□□□□□□□□□□□□□□

□□

□□□□(object) □□□"blob"□"tree"□ "commit" □"tag"□

- **“blob”**□□□□□□□□□□□□□□□□□
- **“tree”**□□□□□□□□□□□□□**“tree”**□□ **“blob”**□□□□□□□□□□
- □□**“commit”**□□□□□"tree"□□commits□□□□□□□
- □□**“tag”**□□□□□□□□□(commit) □□□□

□□□□□Git□□

□SVN□□□

Git□□□□□□□□□□□□□□□□□□□□□□□□□□□Subversion□CVS□Perforce□Mercurial □□□□□□□“*□□□□□□*” □Delta Storage systems□, □□□□□□□□□□□(commit) □□□□□□Git□□□□□□□□□□□□□□□□□□□□□□□□□snapshot□□□□□□□□□□□□□□Git□□□□□□□□□□□

Blob□□

□□blob□□□□□□□□□□□.

□□□□□git show□□□□□□□blob□□□□□□□□□□□□□□□□Blob□□□SHA1□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ git show 6ff87c4664

 Note that the only valid version of the GPL as far as this project
 is concerned is _this_ particular version of the license (ie v2, not
 v2.2 or v3.x or whatever), unless explicitly otherwise stated.
...
```

□□"blob□□□"□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.

□□blob□□blob□□□Blob□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Tree [illegible]

[illegible]tree[illegible](bunch)[illegible]blob[illegible]tree[illegible]

git show[illegible]tree[illegible]git ls-tree[illegible]tree[illegible]SHA1[illegible]

```
$ git ls-tree fb3a8bdd0ce
100644 blob 63c918c667fa005ff12ad89437f2fdc80926e21c    .gitignore
100644 blob 5529b198e8d14decbe4ad99db3f7fb632de0439d    .mailmap
100644 blob 6ff87c4664981e4397625791c8ea3bbb5f2279a3    COPYING
040000 tree 2fb783e477100ce076f6bf57e4a6f026013dc745    Documentation
100755 blob 3c0032cec592a765692234f1cba47dfdcc3a9200    GIT-VERSION-GEN
100644 blob 289b046a443c0647624607d471289b2c7dcd470b    INSTALL
100644 blob 4eb463797adc693dc168b926b6932ff53f17d0b1    Makefile
100644 blob 548142c327a6790ff8821d67c2ee1eff7a656b52    README
...
```

[illegible]tree[illegible](list)[illegible]mode[illegible]SHA1[illegible] [illegible]([illegible])[illegible]

[illegible]tree[illegible](reference): [illegible]blob[illegible], [illegible]tree[illegible]. Tree[illegible]blob[illegible]SHA1[illegible]tree[illegible]Git[illegible]tree[illegible]

([illegible]submodules[illegible]trees[illegible]commits[illegible]. [illegible]**Submodules** [illegible])

[illegible]mode[illegible]644 [illegible] 755[illegible]Git[illegible].

Commit[illegible]

"commit[illegible]"[illegible]"tree[illegible]", [illegible].

[illegible] --pretty=raw [illegible]git show [illegible] git log [illegible](commit):

```
$ git show -s --pretty=raw 2be7fcb476
commit 2be7fcb4764f2dbcee52635b91fedb1b3dcf7ab4
tree fb3a8bdd0ceddd019615af4d57a53f43d8cee2bf
parent 257a84d9d02e90447b149af58b271c19405edb6a
author Dave Watson <dwatson@mimvista.com> 1187576872 -0400
committer Junio C Hamano <gitster@pobox.com> 1187591163 -0700

    Fix misspelling of 'suppress' in docs

    Signed-off-by: Junio C Hamano <gitster@pobox.com>
```

[illegible], [illegible](commit)[illegible]:

- [illegible] **tree**[illegible]: tree[illegible]SHA1[illegible], [illegible].
- [illegible] (parent(s)): [illegible](commit)[illegible]SHA1[illegible]. [illegible]; [illegible](merge commits)[illegible]. [illegible], [illegible]“[illegible]"(root commit), [illegible](revision). [illegible]“[illegible]"(root commit). [illegible]"[illegible]“[illegible]([illegible]).
- [illegible] : [illegible],[illegible].

- □□□□committer): □□□□□□(commit)□□□□□, □□□□□□□□□. TA□□□□□□□□□□□□□; □□□□□□□□□□(patch)□□□□□□□□□□□□, □□□□□□□(commit).

□□□ □□□□□□□□□.

□□: □□□□(commit)□□□□□□□□□□□□□□□□□□□□□□□□; □□□□□(changes)□□□□□□□□(parents)□□□□□□□□□□. □□□□□□, □□git□□□□□□□□□□□□□□□□□□□□□, □□□□□□□□(explicitly)□□□□□□□□□□□.□(□□□□□□ git diff □ -M□□□□□□)

□□□ git commit □□□□□□□(commit), □□□□(commit)□□□□□□□□□□□□(current HEAD),□□□□□□□□□□□□(index)□□□□□□□.

□□□□

□□□□□□□□3□□□□□□□(blob, tree □ commit), □□□□□□□□□□□□□□□□□□□□□□□□□□.

□□□□□□□□□, □□□□□□□□:

```
$>tree
.
|-- README
`-- lib
    |-- inc
    |   `-- tricks.rb
    `-- mylib.rb

2 directories, 3 files
```

□□□□□□□□(commit)□□□Git□□□, □Git□□□□□□□□□□□□□:

□□□□□: □□□□□□□□**tree**□□ (□□□□□), □□□□□□□□□□□□□□**blob**□□ . □□□□□**commit**□□ □□□□tree□□(root of trees), □□□□□□□□□□□□□□□□□□□□.

□□□□

□□□□□□□□□□□□□(□□□:□□SHA1□□), □□□□, □□□, □□□□□□□□("tagger"), □□□□□□□□□□□□(signature)□□□. □□git cat-file □□□□□□□□□□:

```
$ git cat-file tag v1.5.0
object 437b1b20df4b356c9342dac8d38849f24ef44f27
type commit
tag v1.5.0
tagger Junio C Hamano <junkio@cox.net> 1171411200 +0000

GIT 1.5.0
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iD8DBQBF0lGqwMbZpPMRm5oRAuRiAJ9ohBLd7s2kqjkKlq1qqC57SbnmzQCdG4ui
nLE/L9aUXdWeTFPron96DLA=
=2E+0
-----END PGP SIGNATURE-----
```

□□ git tag, □□□□□□□□□□□□□□□□□. (□□:git tag □□□□□□□□□□ "□□□□□□"(lightweight tags), □□□□□□□□□□□, □□□□□ "refs/tags/" □□□□□□□).

GIT□□ □ □□□□

Git□□

'Git□□'□□□□□□□□□□□□□□□□□□ - □□□□□□(commits,trees,blobs,tags), □□□□□□□□□□.

□□□□□□□□□□'Git□□'(□□SVN,CVS□□□□□□□□□□□□□□□),□□□□'.git'□□□□□□□□□□□□□(□□□□□□,□□□□□□□□). □□□□□□□□□□□, □□□□□□□□□□:

```
$>tree -L 1
.
|-- HEAD         # □□git□□□□□□□□□
|-- config       # □□□□□□git config□□□□□
|-- description  # □□□□□□
|-- hooks/       # □□□□□□□□
|-- index        # □□□
|-- logs/        # □□refs□□□□□
|-- objects/     # Git□□□□□□□ (commits, trees, blobs, tags)
`-- refs/        # □□□□□□□□□□□□□(commit)□
```

(□□□□□□□□ □□/□□ □ 'Git□□' □□, □□□□□□□□□)

□□□□

Git□ '□□□□' □□□□□□□□(checkout)□□□□□□□. □□□□□□□□□□□□, □□□□□□□□□□□□□□□. □□□□□□□□□ 'Git□□'□ ;□□□□□□□□□□□□(checkout) □□□□□, □□
□□□□□□□□□□□□□□□□□□□□□□(commit)□□.

□□□: 'Git□□' □□□□□□□□□□□□'.git'□□.

GIT□□

Git□□□□□□□□□□□□□□□□□□□□(staging area). □□□, □□□□□□□□□□□□□□□(commit). □□□□□□□□□(commit), □□□□□□□□□□□□(index)□□□□, □□□□□□□□□□.

□□□□

□□ git status □□□□□□□□□□□□□□. □□□ git status□□, □□□□□: □□□□□□□(□□□□□Git□□□), □□□□□□□□□□□□, □□□□□□□□□□(untracked).

```
$>git status
# On branch master
# Your branch is behind 'origin/master' by 11 commits, and can be fast-forwarded.
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   modified:   daemon.c
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#
#   modified:   grep.c
#   modified:   grep.h
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
```

```
#
#   blametree
#   blametree-init
#   git-gui/git-citool
```

□□□□□□□□□(index), □□□□□□□□□□□□, □□□□□□□□□□□(name of the tree that it described)□□□□□□□□.

□□, □□□□□Git□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□□. □□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□. □□, □□□□□□□□□□□□□, □□□□□Git.

Chapter 2

[illegible]

[illegible]GIT

[illegible]

[illegible]Unix[illegible]Git[illegible]Git Download Page[illegible],[illegible],[illegible]:

```
$ make prefix=/usr all ;# as yourself
$ make prefix=/usr install ;# [illegible]root[illegible]
```

[illegible]: expat,curl, zlib, [illegible] openssl; [illegible]expat [illegible]

Linux

[illegible]Linux[illegible](native package management system)[illegible].

```
$ yum install git-core  #[illegible]redhat[illegible]yum

$ apt-get install git-core  #[illegible]debian, ubuntu[illegible]apt-get
```

[illegible] .deb [illegible] .rpm [illegible]:

RPM Packages

Stable Debs

[illegible]Linux[illegible],[illegible]Article: Installing Git on Ubuntu

Mac 10.4

[illegible]Mac10.4[illegible] 10.5,[illegible]MacPorts,[illegible]MacPorts[illegible]Git[illegible]MacPort, [illegible].

[illegible]MacPorts[illegible]:

```
$ sudo port install git-core
```

[illegible]:

Article: Installing Git on Tiger

Article: Installing Git and git-svn on Tiger from source

Mac 10.5

[illegible]Leopard[illegible]MacPorts[illegible],[illegible]:"[illegible]", [illegible]Git OSX Installer

[illegible]:

Article: Installing Git on OSX Leopard

Article: Installing Git on OS 10.5

Windows

[illegible]Windows[illegible]Git[illegible]msysGit[illegible]

Git on Windows [illegible]"screencast"[illegible]windows[illegible]Git.

[illegible]

Git [illegible]

[illegible]Git[illegible]email,[illegible]commit[illegible]

```
$ git config --global user.name "Scott Chacon"
$ git config --global user.email "schacon@gmail.com"
```

[illegible],[illegible](home directory)[illegible]*~/.gitconfig* [illegible]. [illegible]:

```
[user]
        name = Scott Chacon
        email = schacon@gmail.com
```

[illegible]:[illegible],[illegible].

[illegible]([illegible]);[illegible]git config [illegible]*--global* [illegible]. [illegible]*.git/config* [illegible][user][illegible]([illegible]).

Chapter 3

[illegible]

[illegible]GIT[illegible]

[illegible]Git[illegible]Git[illegible]*clone* ([illegible])[illegible]

Clone[illegible]

[illegible](copy),[illegible](Git URL). Git[illegible]Git URL[illegible]ssh://, http(s)://, git://,[illegible]git [illegible]ssh [illegible]. [illegible]Git[illegible] git:// [illegible]

```
git clone git://git.kernel.org/pub/scm/git/git.git
```

[illegible]http [illegible]:

```
git clone http://www.kernel.org/pub/scm/git/git.git
```

git://[illegible],[illegible]http[illegible],[illegible]http[illegible].[illegible],[illegible]: 'git',[illegible]Git[illegible].

[illegible]Git[illegible]"Git URL"[illegible]'.git'[illegible],[illegible](clone)[illegible]: ([illegible]. *git clone http://git.kernel.org/linux/kernel/git/torvalds/linux-2.6.git* [illegible]'linux-2.6')

[illegible]

[illegible]"project.tar.gz"[illegible]Git[illegible].

```
$ tar xzf project.tar.gz
$ cd project
$ git init
```

Git[illegible]:

```
Initialized empty Git repository in .git/
```

[illegible]project[illegible]".git" [illegible]

gitcast:c1_init

□□□□□□□

□□□□□□□□□□□□□□□□□□□□.

```
$ git add file1 file2 file3
```

□□□□commit□□□□□□□□□□□git diff □□□□□ --cached □□ ,□□□□□□□□□□(commit)□

```
$ git diff --cached
```

(□□□□--cached□□□git diff □□□□□□□□□□□□□□□□□□□□□□□.) □□□□□git status□□□□□□□□□□□□□□:

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   modified:   file1
#   modified:   file2
#   modified:   file3
#
```

□□□□□□□□□□□□, □□□□□, □□□□□□□□□□□□□□□□□□. □□□□□□□□

```
$ git commit
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.

□□□git add □□□□□□□□

```
$ git commit -a
```

□□□□□□□□□□□□□□□□(□□□□□□□□□□)□□□□□□□□□□□□□□□□□□□

□□□□□□□□commit□□□□□□□□□□:commit□□□□□□□□□□□□□□□□□□□□□□□□□commit□□□□□(□□□□□□50□□□)□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□commit□□
email□□□□□□□□□□□□□□□□□□email□□□.

Git□□□□□□□□□□

□□□□□□□□□□□□□□ "add" □□□□□□□□□□□□□□□□□□□□□□□□Git□□ ”add” □□□□□□□□□□□□□□□.git add □□□; □□□
□□□□, Git□□□□□□□□□□□□□□□□□□(stage)□□□□□□□□□commit□□□□□

gitcast:c2_normal_workflow

□□□□□@□□

□□Git□□□□□□□□□□□□□□□□□□□□□□□□”experimental”□□□□

```
$ git branch experimental
```

□□□□□□□□□□□□

```
$ git branch
```

□□□□□□□□□□□□

```
  experimental
* master
```

“experimental” □□□□□□“master”□□Git□□□□□□□□(“*”)□□□□□□□□□□□

```
$ git checkout experimental
```

□□”experimental”□□□□□□□□□□□(commit)□□□□□ “master”□□

```
(edit file)
$ git commit -a
$ git checkout master
```

□□□□□□□□“experimental”□□□□□□□□□□□□□□“master”□□□□□□□□□□□

□□□□“master”□□□□□□□:

```
(edit file)
$ git commit -a
```

□□□□□□□□□□□(diverged)□□□□□□□□□□“experimental”□“master”□□□:

```
$ git merge experimental
```

□□□□□□□□□□(conflict), □□□□□□□□□□□□□□□□□□□□□□□□□:

```
$ git diff
```

□□□□□□□□□□□□□□□□

```
$ git commit -a
```

□□(commit)□□□□□□□□:

```
$ gitk
```

□□gitk□□□□□□□□□□□□□

□□□□□□□ “experimental” □□(□□□□)□

```
$ git branch -d experimental
```

git branch -d□□□□□□□□□□□□□□□□□□□□□□□git branch –D□□□□□□□□□“crazy-idea”□□

```
$ git branch -D crazy-idea
```

□□□□□□□□□□□□□□□

□□

□□□□□□□□□□□□git merge:

```
$ git merge branchname
```

□□□□□□□"branchname"□□□□□□□□□□□□□□□□(□□--□□

```
$ git merge next
 100% (4/4) done
Auto-merged file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□(index)□□□□git commit□□□□□□□□□□□□□□□□□ □□□

□□□□gitk□□□commit□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

□□□□□□□□

□□□□□□□□□□□□□□□git□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□

□□□(conflicts)□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□git commit□□□□:

```
$ git commit
file.txt: needs merge
```

□□□□ git status □□□□□□□□□□□(unmerged),□□□□□□□□□□□□□□□□□□□□□□:

```
<<<<<<< HEAD:file.txt
Hello world
=======
Goodbye
>>>>>>> 77976da35a11db4580b80ae27e8d65caf5208086:file.txt
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□:

```
$ git add file.txt
$ git commit
```

□□

□□□□□□□□□□□□□□□□□□□□□□□git□□□□□□□□□□ □□□□□□□

□□□□□□

□□□

```
$ git reset --hard HEAD
```

□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ git reset --hard ORIG_HEAD
```

□□ □□□□□□□□□□□□□□□

□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□(commit), □□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□(commit)□□□□□□□□□□□git □□□□□□“□□□□"(fast forward)□□□git □□□□□□□□□(commit),□□□□□□□□□□
□□□□□□

gitcast:c6-branch-merge

□□□□ □GIT□□

git log□□□□□□□□□□□(commit)□ ......

```
$ git log v2.5..        # commits since (not reachable from) v2.5
$ git log test..master  # commits reachable from master but not test
$ git log master..test  # commits reachable from test but not master
$ git log master...test # commits reachable from either test or
                        #    master, but not both
$ git log --since="2 weeks ago" # commits from the last 2 weeks
$ git log Makefile      # commits that modify Makefile
$ git log fs/           # commits that modify any file under fs/
$ git log -S'foo()'     # commits that add or remove any file data
                        # matching the string 'foo()'
$ git log --no-merges   # dont show merge commits
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□"v2.5“□ □□fs□□□□□□Makefile□□□.

```
$ git log v2.5.. Makefile fs/
```

Git□□□git log□□□□□□□□□□□□□□□□□(commit)□

```
commit f491239170cb1463c7c3cd970862d6de636ba787
Author: Matt McCutchen <matt@mattmccutchen.net>
Date:   Thu Aug 14 13:37:41 2008 -0400

    git format-patch documentation: clarify what --cover-letter does

commit 7950659dc9ef7f2b50b18010622299c508bfdfc3
Author: Eric Raible <raible@gmail.com>
Date:   Thu Aug 14 10:12:54 2008 -0700

    bash completion: 'git apply' should use 'fix' not 'strip'
    Bring completion up to date with the man page.
```

□□□□□git log□□□□(patchs):

```
$ git log -p

commit da9973c6f9600d90e64aac647f3ed22dfd692f70
Author: Robert Schiele <rschiele@gmail.com>
Date:   Mon Aug 18 16:17:04 2008 +0200

    adapt git-cvsserver manpage to dash-free syntax

diff --git a/Documentation/git-cvsserver.txt b/Documentation/git-cvsserver.txt
index c2d3c90..785779e 100644
--- a/Documentation/git-cvsserver.txt
+++ b/Documentation/git-cvsserver.txt
@@ -11,7 +11,7 @@ SYNOPSIS
 SSH:
```

```
 [verse]
-export CVS_SERVER=git-cvsserver
+export CVS_SERVER="git cvsserver"
 'cvs' -d :ext:user@server/path/repo.git co <HEAD_name>

 pserver (/etc/inetd.conf):
```

□□□□

□□□--stat□□□□'git log',□□□□□□□□□(commit)□□□□□□□□□, □□□□□□□□□□□□□□□□□.

```
$ git log --stat

commit dba9194a49452b5f093b96872e19c91b50e526aa
Author: Junio C Hamano <gitster@pobox.com>
Date:   Sun Aug 17 15:44:11 2008 -0700

    Start 1.6.0.X maintenance series

 Documentation/RelNotes-1.6.0.1.txt |   15 +++++++++++++++
 RelNotes                           |    2 +-
 2 files changed, 16 insertions(+), 1 deletions(-)
```

□□□□□

□□□□□□□□□□□□□□□□□‘--pretty'□□□□□□□□□□□□□□‘oneline':

```
$ git log --pretty=oneline
a6b444f570558a5f31ab508dc2a24dc34773825f dammit, this is the second time this has reverted
49d77f72783e4e9f12d1bbcacc45e7a15c800240 modified index to create refs/heads if it is not
9764edd90cf9a423c9698a2f1e814f16f0111238 Add diff-lcs dependency
e1ba1e3ca83d53a2f16b39c453fad33380f8d1cc Add dependency for Open4
0f87b4d9020fff756c18323106b3fd4e2f422135 merged recent changes: * accepts relative alt pat
f0ce7d5979dfb0f415799d086e14a8d2f9653300 updated the Manifest file
```

□□□□□□□□ 'short' □□:

```
$ git log --pretty=short
commit a6b444f570558a5f31ab508dc2a24dc34773825f
Author: Scott Chacon <schacon@gmail.com>

    dammit, this is the second time this has reverted

commit 49d77f72783e4e9f12d1bbcacc45e7a15c800240
Author: Scott Chacon <schacon@gmail.com>

    modified index to create refs/heads if it is not there

commit 9764edd90cf9a423c9698a2f1e814f16f0111238
Author: Hans Engel <engel@engel.uk.to>

    Add diff-lcs dependency
```

□□□□‘medium','full','fuller','email' □‘raw'. □□□□□□□□□□□□□□□□ □□□□□‘--pretty=format'□□(□□git log)□□□□□□□"□□“.

```
$ git log --pretty=format:'%h was %an, %ar, message: %s'
a6b444f was Scott Chacon, 5 days ago, message: dammit, this is the second time this has re
49d77f7 was Scott Chacon, 8 days ago, message: modified index to create refs/heads if it i
```

```
9764edd was Hans Engel, 11 days ago, message: Add diff-lcs dependency
e1ba1e3 was Hans Engel, 11 days ago, message: Add dependency for Open4
0f87b4d was Scott Chacon, 12 days ago, message: merged recent changes:
```

□□□□□□□□□□□□□□'--graph'□□□□□□□□□□□□(commit graph),□□□□□□:

```
$ git log --pretty=format:'%h : %s' --graph
* 2d3acf9 : ignore errors from SIGCHLD on trap
*   5e3ee11 : Merge branch 'master' of git://github.com/dustin/grit
|\
| * 420eac9 : Added a method for getting the current branch.
* | 30e367c : timeout code and tests
* | 5a09431 : add timeout protection to grit
* | e1193f8 : support for heads with slashes in them
|/
* d6016bc : require time for xmlschema
```

□□□ASCII□□□□□□□□□□□□□□□□(commit history)□□

□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□git□□□□□□□□(commit)□□□□□□□□□□□□□□□□□□□□□□git□□□□□□□□□□□□□□□□□□□□□□□□□(commits)□□□□□□□□□□

□□□□□□□□□□□□□□□□□□git log□□□□□□□□(ordering option).

□□□□□□□□(commits)□□□□□(reverse chronological)□□□□□

□□□□□□□□‘--topo-order'□□□□□□□□□(commits)□□□□□□□□(□□□□□□□□□□□□□□□□). □□□□git log□□□□□□□□□□□git□□□□□□□□□□□□□“□□□"(development lines)□
□□□□□□.

```
$ git log --pretty=format:'%h : %s' --topo-order --graph
*   4a904d7 : Merge branch 'idx2'
|\
| *   dfeffce : merged in bryces changes and fixed some testing issues
| |\
| | * 23f4ecf : Clarify how to get a full count out of Repo#commits
| | *   9d6d250 : Appropriate time-zone test fix from halorgium
| | |\
| | | * cec36f7 : Fix the to_hash test to run in US/Pacific time
| | * | decfe7b : fixed manifest and grit.rb to make correct gemspec
| | * | cd27d57 : added lib/grit/commit_stats.rb to the big list o' files
| | * | 823a9d9 : cleared out errors by adding in Grit::Git#run method
| | * |   4eb3bf0 : resolved merge conflicts, hopefully amicably
| | |\ \
| | | * | d065e76 : empty commit to push project to runcoderun
| | | * | 3fa3284 : whitespace
| | | * | d01cffd : whitespace
| | | * | 7c74272 : oops, update version here too
| | | * | 13f8cc3 : push 0.8.3
| | | * | 06bae5a : capture stderr and log it if debug is true when running commands
| | | * | 0b5bedf : update history
| | | * | d40e1f0 : some docs
| | | * | ef8a23c : update gemspec to include the newly added files to manifest
| | | * | 15dd347 : add missing files to manifest; add grit test
| | | * | 3dabb6a : allow sending debug messages to a user defined logger if provided; tes
| | | * | eac1c37 : pull out the date in this assertion and compare as xmlschemaw, to avoi
| | | * | 0a7d387 : Removed debug print.
| | | * | 4d6b69c : Fixed to close opened file description.
```

[illegible]'--date-order'[illegible]. [illegible]'--topo-order'[illegible]"[illegible]"(developmen
lines)[illegible](parallel development)[illegible]

```
$ git log --pretty=format:'%h : %s' --date-order --graph
*   4a904d7 : Merge branch 'idx2'
|\
* | 81a3e0d : updated packfile code to recognize index v2
| *   dfeffce : merged in bryces changes and fixed some testing issues
| |\
| * | c615d80 : fixed a log issue
|/ /
| * 23f4ecf : Clarify how to get a full count out of Repo#commits
| *   9d6d250 : Appropriate time-zone test fix from halorgium
| |\
| * | decfe7b : fixed manifest and grit.rb to make correct gemspec
| * | cd27d57 : added lib/grit/commit_stats.rb to the big list o' file
| * | 823a9d9 : cleared out errors by adding in Grit::Git#run method
| * |   4eb3bf0 : resolved merge conflicts, hopefully amicably
| |\ \
| * | | ba23640 : Fix CommitDb errors in test (was this the right fix?
| * | | 4d8873e : test_commit no longer fails if you're not in PDT
| * | | b3285ad : Use the appropriate method to find a first occurrenc
| * | | 44dda6c : more cleanly accept separate options for initializin
| * | | 839ba9f : needed to be able to ask Repo.new to work with a bar
| | * | d065e76 : empty commit to push project to runcoderun
* | | | 791ec6b : updated grit gemspec
* | | | 756a947 : including code from github updates
| | * | 3fa3284 : whitespace
| | * | d01cffd : whitespace
| * | | a0e4a3d : updated grit gemspec
| * | | 7569d0d : including code from github updates
```

[illegible] '--reverse'[illegible]

gitcast:c4-git-log

[illegible] - GIT DIFF

[illegible] git diff [illegible]

```
$ git diff master..test
```

[illegible]'master','test'[illegible] [illegible]'test'[illegible]3[illegible]'.'[illegible]'.' [illegible]

```
$ git diff master...test
```

git diff [illegible] [illegible]

[illegible](commit)

[illegible]git diff[illegible]

```
$ git diff
```

[illegible] staged([illegible])[illegible] [illegible]

[illegible](staged,[illegible]),[illegible]

```
$ git diff --cached
```

[illegible]"-a"[illegible] "git commit"[illegible]

```
$ git diff HEAD
```

[illegible] [illegible]"git commit -a"[illegible]

[illegible]

[illegible]:

```
$ git diff test
```

[illegible]'test'[illegible] [illegible]

```
$ git diff HEAD -- ./lib
```

[illegible]lib[illegible]([illegible] [illegible])[illegible]

[illegible] [illegible]'--stat' [illegible]

```
$>git diff --stat
 layout/book_index_template.html                    |    8 ++-
 text/05_Installing_Git/0_Source.markdown           |   14 ++++++
 text/05_Installing_Git/1_Linux.markdown            |   17 +++++++
 text/05_Installing_Git/2_Mac_104.markdown          |   11 +++++
 text/05_Installing_Git/3_Mac_105.markdown          |    8 ++++
 text/05_Installing_Git/4_Windows.markdown          |    7 +++
 .../1_Getting_a_Git_Repo.markdown                  |    7 +++-
 .../0_ Comparing_Commits_Git_Diff.markdown         |   45 +++++++++++++++++++-
 .../0_ Hosting_Git_gitweb_repoorcz_github.markdown |    4 +-
 9 files changed, 115 insertions(+), 6 deletions(-)
```

[illegible]

[illegible]

[illegible]Alice[illegible]/home/alice/project[illegible]git [illegible](repository)[illegible]Bob[illegible]

Bob [illegible]:

```
$ git clone /home/alice/project myrepo
```

[illegible]"myrepo"[illegible]Alice[illegible] [illegible](clone). [illegible] [illegible]

Bob [illegible](commit)[illegible]:

```
(edit files)
$ git commit -a
(repeat as necessary)
```

□□□□□□□□□□Alice□□□/home/bob/myrepo□□□□□□□□ (pull)□□□□□□□□□□□□□:

```
$ cd /home/alice/project
$ git pull /home/bob/myrepo master
```

□□□Bob□□(master)□□□□□□Alice□□□□□□□□□□Alice□ Bob□□□□□□□□□□□□□□□□□□□□□□□□□□□□□. (□□□"master"□□□□□□□□□□□□□□□□□□□□□□□□ □□□□)

git pull□□□□□□□□: □□□□□□(remote branch)□□□□ □□□□□□□□□□□□□□□□□

□□□□□□□□□□□□(remote branch),□□□□□□□□□□:

```
$ git remote add bob /home/bob/myrepo
```

□□□Alic□□□"git fetch"□□□"git pull"□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□

```
$ git fetch bob
```

□□□git remote□□□□□Bob□□□□□□□□□□□□(□□) □□□□Bob□□□□□□□□□□□□□□□□□□□□□□□□ □□□bob/master.

```
$ git log -p master..bob/master
```

□□□□□□Bob□Alice□□□□(master)□□□□□□□□□□□□□□□□

□□□□□□□,Alice□□□□□□□□□□□□□□□□

```
$ git merge bob/master
```

□□□□(merge)□□□□pull□□□□□□□□□□□□□□

```
$ git pull . remotes/bob/master
```

□□□git pull □□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□

□□□Bob□□□□□□□□□--□Alice□□□□□□□(pull):

```
$ git pull
```

□□Bob□Alice□□□□□(clone)□□□□□□□□□□Alice□□□□ □□□□Git□Alice□□□□□□□□Bob□□□□□□□□□□□□□□ □git pull□□□□

```
$ git config --get remote.origin.url
/home/alice/project
```

(□□□□□git clone□□□□□□□□□□□□□□□"git config -l",git config □□□□□□□□□□□□□□□□.)

Git□□□□□□□□□□(pristine)□Alice□□□(master)□□ "origin/master"□□□

```
$ git branch -r
  origin/master
```

□□Bob□□□□□□□□□□□□□□□□□□□□ssh□□□□□"clone" □"pull"□□□

```
$ git clone alice.org:/home/alice/project myrepo
```

git□□□□□□□(native protocol),□□□□□rsync□http; □□□□ □□git pull □□□□□□□□□□

Git□□□□CVS□□□□□□□□□□□□□□□□□□□□□□□(push) □□□□□□□□□□□□□□□□git push gitcvs-migration.

□□Git□□

□□□□□□□□□□□□□□□□□□□□□□(maintainer)□git pull □□□□□□□□□□□□□□□□□□□□□□"□□□□□□□□□□□□ □□□□□□□□□□□

□□□□□□□(maintainer)□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□(pull)□□□□□□git□□□□□□□□□□□□□□ □□□□□□□

```
$ git clone /path/to/repository
$ git pull /path/to/other/repository
```

□□□□□□ssh□□□

```
$ git clone ssh://yourhost/~you/repository
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□(public repository). □□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□(push) □□□□□□□□□□□□□□□□□□□□□□□□□□(pull)□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□

```
                      you push
   your personal repo ------------------> your public repo
     ^                                     |
     |                                    |
     | you pull                            | they pull
     |                                    |
     |                                    |
        |               they push            V
   their public repo <------------------- their repo
```

□□□□□□□□□□□

□□http□□git□□□□□□□□□□□□(fetch)□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□: □□http□WebDav□□□□□□□□□□□,□□□□□□git over http.

□□□□□□□□□git push□□ □ssh□□; □□□□□"master" □□□□□□□□□"master"□□□□□□□□□□□:

```
$ git push ssh://yourserver.com/~you/proj.git master:master
```

or just

□□:

```
$ git push ssh://yourserver.com/~you/proj.git master
```

□git-fetch□□□□git-push□□□□□□□□□□□"□□□□"(fast forward) □□□□; □□□□□□□□□□□□□□□.

□(push)□□□□□□□□□□□□□□(bare respository). □□□□□□□ □□□□□□□□(checked-out working tree)□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□ □□□□□□□□□□□

□□git-fetch□□□□□□□□□□□□□□□□□□□□:)□

□□□□□□□:

```
$ cat >>.git/config <<EOF
```

```
[remote "public-repo"]
    url = ssh://yourserver.com/~you/proj.git
EOF
```

[illegible]:

```
$ git push public-repo master
```

[illegible]: git config[illegible]remote..url, branch..remote, [illegible]remote..push[illegible].

[illegible]

[illegible](push)[illegible]"[illegible]"(fast forward),[illegible] [illegible]

```
error: remote 'refs/heads/master' is not an ancestor of
local  'refs/heads/master'.
Maybe you are not up-to-date and need to pull first?
error: failed to push to 'ssh://yourserver.com/~you/proj.git'
```

[illegible]

```
- [illegible] `git-reset --hard` [illegible]

- [illegible] `git-commit --amend` [illegible]

- [illegible] `git-rebase` [illegible]rebase[illegible].[illegible]
```

[illegible]git-push[illegible]

```
$ git push ssh://yourserver.com/~you/proj.git +master
```

[illegible](commit) [illegible](commit)[illegible]

[illegible] [illegible]

[illegible](push)[illegible](repository)[illegible] [illegible]"pull"[illegible]"fetch"[illegible]"rebase" [illegible](push)[illegible]
[illegible] gitcvs-migration[illegible]

gitcast:c8-dist-workflow

GIT[illegible]

[illegible]

[illegible] git tag[illegible](tag)[illegible](commit):

```
$ git tag stable-1 1b2e1d63ff
```

[illegible]stable-1 [illegible](commit) "1b2e1d63ff" [illegible](refer)[illegible]

[illegible]“[illegible]"[illegible]

[illegible](tag)[illegible](sign it cryptographically), [illegible] ”[illegible]".

[illegible]

[illegible] "-a", "-s" [illegible] "-u" [illegible] [illegible](tag message). [illegible]"-m"[illegible] "-F" [illegible](tag message).

[illegible](commit comment)[illegible]

[illegible]Git[illegible] [illegible](commit). [illegible] [illegible](sign), [illegible](commi

[illegible]:

```
$ git tag -a stable-1 1b2e1d63ff
```

[illegible](commit). ([illegible]Linux[illegible] [illegible](tree),[illegible](commit)).

[illegible]

[illegible]GPG key,[illegible].[illegible].*git/config [illegible]* ~.gitconfig[illegible]key.

[illegible]:

```
[user]
    signingkey = <gpg-key-id>
```

[illegible]:

```
$ git config (--global) user.signingkey <gpg-key-id>
```

[illegible]"-s" [illegible]“[illegible]”[illegible]

```
$ git tag -s stable-1 1b2e1d63ff
```

[illegible]GPG key,[illegible]"-u“ [illegible]

```
$ git tag -u <gpg-key-id> stable-1 1b2e1d63ff
```

Chapter 4

[illegible]

[illegible]

[illegible]Git[illegible](track)[illegible] [illegible](track)[illegible] [illegible]"git add"[illegible] [illegible]
[illegible] [illegible](untracked)[illegible]; [illegible]"git add ." [illegible]"git commit -a" [illegible]"git status"[illegible]

[illegible]".gitignore"[illegible]Git[illegible] [illegible]:

```
# [illegible]'#' [illegible].
# [illegible] foo.txt [illegible].
foo.txt
# [illegible] html [illegible],
*.html
# foo.html[illegible].
!foo.html
#  [illegible].o [illegible] .a[illegible].
*.[oa]
```

[illegible]gitignore [illegible]. [illegible]".gitignore" [illegible](working tree)[illegible](ignore) [illegible].gitignore[illegible]
[illegible]([illegible] git add .gitignore [illegible] git commit[illegible]), [illegible] [illegible]

[illegible],[illegible] .git/info/exclude[illegible]Git[illegible]core.excludesfile[illegible] [illegible]Git[illegible]
[illegible]:gitignore [illegible]

REBASE

[illegible]"origin"[illegible]"mywork"[illegible]

```
$ git checkout -b mywork origin
```

[illegible](commit).

```
$ vi file.txt
$ git commit
$ vi otherfile.txt
$ git commit
```

```
...
```

[illegible]"origin"[illegible]. [illegible]"origin"[illegible]"mywork"[illegible]"[illegible]"[illegible]"[illegible]"[illegible]

[illegible]"pull"[illegible]"origin"[illegible] [illegible]"[illegible]"(merge commit):

[illegible]"mywork"[illegible]git rebase:

```
$ git checkout mywork
$ git rebase origin
```

[illegible]"mywork"[illegible](commit)[illegible] [illegible](patch)([illegible]".git/rebase"[illegible]),[illegible]"mywork"[illegible] [illegible]"origin"[illegible]"mywork"[illegible]

[illegible]'mywork'[illegible](commit),[illegible] [illegible](pruning garbage collection), [illegible]. [illegible]git gc)

[illegible](merge)[illegible]rebase[illegible]

[illegible]rebase[illegible](conflict). [illegible]Git[illegible]rebase[illegible] [illegible]"git-add"[illegible](index), [illegible] git-commit,[illegible]:

```
$ git rebase --continue
```

[illegible]git[illegible](apply)[illegible]

[illegible]`--abort`[illegible]rebase[illegible]"mywork" [illegible]rebase[illegible]

```
$ git rebase --abort
```

gitcast:c7-rebase

[illegible]REBASE

[illegible]rebase[illegible]rebase[illegible]rebase[illegible]

[illegible]rebase[illegible]'git rebase'[illegible]'-i'[illegible]'--interactive'[illegible]

```
$ git rebase -i origin/master
```

[illegible]rebase[illegible]origin[illegible]origin[illegible]

[illegible]rebase[illegible]log[illegible]

```
$ git log github/master..
```

[illegible]'rebase -i'[illegible]

```
pick fc62e55 added file_size
pick 9824bf4 fixed little thing
pick 21d80a5 added number to log
pick 76b9da6 added the apply command
pick c264051 Revert "added file_size" - not implemented correctly

# Rebase f408319..b04dc3d onto f408319
#
# Commands:
#  p, pick = use commit
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

[illegible]5[illegible]

```
(action) (partial-sha) (short commit message)
```

[illegible]action[illegible]'edit'[illegible]'squash'[illegible]'pick'[illegible]git[illegible]
[illegible]action[illegible]

[illegible]'pick'[illegible]git[illegible]commit message[illegible]

[illegible]'squash'[illegible]git[illegible]

```
pick   fc62e55 added file_size
squash 9824bf4 fixed little thing
squash 21d80a5 added number to log
squash 76b9da6 added the apply command
squash c264051 Revert "added file_size" - not implemented correctly
```

[illegible]

```
# This is a combination of 5 commits.
# The first commit's message is:
added file_size

# This is the 2nd commit message:

fixed little thing

# This is the 3rd commit message:

added number to log

# This is the 4th commit message:

added the apply command

# This is the 5th commit message:

Revert "added file_size" - not implemented correctly
```

```
This reverts commit fc62e5543b195f18391886b9f663d5a7eca38e84.
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□'edit'□□□git□□

□□□□□□□□□□□□□□□□□□□□□□'edit'□□□

```
pick   fc62e55 added file_size
pick   9824bf4 fixed little thing
edit   21d80a5 added number to log
pick   76b9da6 added the apply command
pick   c264051 Revert "added file_size" - not implemented correctly
```

□□□□□□□□□□□□revert□□□□□□□□□□□□□□□□□□□□□□□□□□21d80a5□□□□□□□□file1□file2□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ git reset HEAD^
$ git add file1
$ git commit 'first part of split commit'
$ git add file2
$ git commit 'second part of split commit'
$ git rebase --continue
```

□□□□6□□□□□□□□5□□

□□□rebase□□□□□□□□□□□□□□□□□□□□□□□□□'pick'□'squash'□'edit'□□□□□□□git□□□□□□□□□□□

□□□□□

□□□□□□□□□□□□□□□Git□□□index□□□□□□□□□□□□□□□□□□□□□□□'git add -i'□□□□□□□□□Git□□□□□□□□□□□□□□□□□□

```
$>git add -i
           staged     unstaged path
  1:    unchanged        +4/-0 assets/stylesheets/style.css
  2:    unchanged      +23/-11 layout/book_index_template.html
  3:    unchanged        +7/-7 layout/chapter_template.html
  4:    unchanged        +3/-3 script/pdf.rb
  5:    unchanged      +121/-0 text/14_Interactive_Rebasing/0_ Interactive_Rebasing.markdown

*** Commands ***
  1: status   2: update   3: revert   4: add untracked
  5: patch    6: diff     7: quit     8: help
What now>
```

□□□□□□□□□□□□□□5□□□□□□□□□□□□□□□□□□unstaged□□

□□□□□□□□□stage□□□□□□□□□□□□'2'□□'u'□□□□□update□□□□□□□□□□□□□□□□□□□□□□□□1-4□□□□□□□□□□□□□□□□□

```
What now> 2
           staged     unstaged path
  1:    unchanged        +4/-0 assets/stylesheets/style.css
  2:    unchanged      +23/-11 layout/book_index_template.html
  3:    unchanged        +7/-7 layout/chapter_template.html
  4:    unchanged        +3/-3 script/pdf.rb
  5:    unchanged      +121/-0 text/14_Interactive_Rebasing/0_ Interactive_Rebasing.markdown
Update>> 1-4
```

```
          staged     unstaged path
* 1:    unchanged        +4/-0 assets/stylesheets/style.css
* 2:    unchanged      +23/-11 layout/book_index_template.html
* 3:    unchanged        +7/-7 layout/chapter_template.html
* 4:    unchanged        +3/-3 script/pdf.rb
  5:    unchanged      +121/-0 text/14_Interactive_Rebasing/0_ Interactive_Rebasing.markdown
Update>>
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
What now> status
          staged     unstaged path
  1:        +4/-0      nothing assets/stylesheets/style.css
  2:      +23/-11      nothing layout/book_index_template.html
  3:        +7/-7      nothing layout/chapter_template.html
  4:        +3/-3      nothing script/pdf.rb
  5:    unchanged      +121/-0 text/14_Interactive_Rebasing/0_ Interactive_Rebasing.markdown
```

□□□□□□□4□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□'git status'□□□□

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   modified:   assets/stylesheets/style.css
#   modified:   layout/book_index_template.html
#   modified:   layout/chapter_template.html
#   modified:   script/pdf.rb
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#
#   modified:   text/14_Interactive_Rebasing/0_ Interactive_Rebasing.markdown
#
```

□□□□□□□□□□□□□□□□□□3: revert□□□□□□□□4: add untracked□□□□□6: diff□□□□□□□□□□□□“□”□□□□□□□□staging patches□□5: patch□□

□□□□'5'□□'p'□git□□□
□□□□□□□□□□□□'git add -i'□□□□□□□□

□□□□book_index_template.html□□□□□□□□□□□

```
        staged     unstaged path
1:        +4/-0      nothing assets/stylesheets/style.css
2:       +20/-7        +3/-4 layout/book_index_template.html
3:        +7/-7      nothing layout/chapter_template.html
4:        +3/-3      nothing script/pdf.rb
5:    unchanged      +121/-0 text/14_Interactive_Rebasing/0_ Interactive_Rebasing.markdown
6:    unchanged       +85/-0 text/15_Interactive_Adding/0_ Interactive_Adding.markdown
```

□□□'git add -i'□□□□□□□□□□□□□7: quit□□□'git commit'□□□□□□□□□□'git commit -a'□□□□□□□□□□□□□□□□□□□□□□□□

gitcast:c3_add_interactive

□□

□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□bug. □□□□□□□□bug□□□□□□, □□□□git stash □□□□□□□□□, □□□□□bug□,□□'□□□'(unstash)□□□□□□□□□□□□.

```
$ git stash "work in progress for foo feature"
```

□□□□□□□□□□□□□□□□□(stash)□, □□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□.

□□, □□□□□□□□□□□□□□.

```
... edit and test ...
$ git commit -a -m "blorpl: typofix"
```

□□□□□bug□, □□□□git stash apply□□□□□□□□□□.

```
$ git stash apply
```

□□□□

□□□□□□□'git stash'□□,□□□□□□□□□□□□□□□□□'□□'(stash)□□□□□□□. □'git stash list'□□□□□□□□□□'□□'(stashes):

```
$>git stash list
stash@{0}: WIP on book: 51bea1d... fixed images
stash@{1}: WIP on master: 9705ae6... changed the browse code to the official repo
```

□□□□□'git stash apply stash@{1}'□□□□□□□□□□□□□□'□□'(stashes). 'git stash clear'□□□□□□□□□□.

GIT□□

□□40□□□□SHA□□□□□□□(commit)□□□□git□□, □□□□□□□□□□.□□Git□,□□□□□'□□'(treeish).

□□□:□□□□□□□□□□□□□,□□□'□□'.

Sha□□

□□□□□□□(commit)□sha□□□ '980e3ccdaac54a0d4de358f3fe5d718027d96aae', git□□□□□□□□□□□:

```
980e3ccdaac54a0d4de358f3fe5d718027d96aae
980e3ccdaac54a0d4
980e3cc
```

□□□□'sha□□'(Partial Sha)□□□□(unique)□□□□□□□□□□(□□□□□5□□□□□□□□□□□)□git□□□'sha□□'(Partial Sha)□□□.

□□, Remote □ □□

□□□□□□,remote□□□□□□SHA□□, □□□□□□□□□□□. □□□master□□□□□□(commit):'980e3'□, □□□□□(push)□origin□□□□□□□'v1.0', □□□□□□git
□□□□:

```
980e3ccdaac54a0d4de358f3fe5d718027d96aae
origin/master
refs/remotes/origin/master
master
refs/heads/master
v1.0
```

```
refs/tags/v1.0
```

□□□□□□□□□□□□□□□□□□□□□:

```
$ git log master

$ git log refs/tags/v1.0
```

□□□□□

The Ref Log that git keeps will allow you to do some relative stuff locally, such as:

Git□□□□□(Ref Log)□□□□□□□‘□□'□□□□:

```
master@{yesterday}

master@{1 month ago}
```

□□□□□□□□□:'master□□□□□□□(head)□□□‘. □□: □□□□□□□□master□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□,□□□□□□□□□□□□□□□.

□□□:□□□□□□□□□□□□□□□□□□□□□.

□□□□□

□□□□□□□□□□□□□□N□□□(ref).

```
master@{5}
```

□□□□□□□□□master□□□□5□□□(ref).

□□□□□

□□□□□□□□□□□N□□□□□□(parent). □□□□□□□□□(merge commits)□□□□□, □□□□□□□□□□(commit object)□□□□□□□□□□(direct parent).

□□□:□□master□□a□b□□□□□□□,□□`master^1` □□□□a, `master^2` □□□□□b.

```
master^2
```

□□□

□□□□□□□□□□□□□(commit object)□□N□□(□)□□□(Nth grandparent). □□:

```
master~2
```

□□□master□□□□□□□□□□□□□□□□□□□□□□(□□:□□□□□□□□□□□:)). □□□□□□□□□□□□□□:

```
master^^
```

□□□□□□□‘□□□'(spec)□□□□, □□□□3□□□□□□□□□□□□□(commit):

```
master^^^^^^
master~3^~2
```

```
master~6
```

□□□□□

□□□□□□□Git□□□□□□□□□□, □□□□□□□(commit object)□□□□□□□□(tree object)□. □□□□□□□□□□□□□(commit object)□□□□□□(tree object)□sha□□, □□□□□‘□
□'□□□□□'^{tree}'□□□□:

```
master^{tree}
```

□□□□□□

□□□□□□□□□□□(blob)□sha□□,□□□□'□□'(treeish)□□□□□□□□(blob)□□□□□□□□□□□□.

```
master:/path/to/file
```

□□

□□, □□□□".."□□□□□□(commit)□□□□□. □□□□□□□□□□"7b593b5" □"51bea1"□□□□"7b593b5□"□□□□□(commit)(□□:51bea1□□□□□□).

```
7b593b5..51bea1
```

□□□□□□□□7b593b□□□□□(commit). □□□: □□□ 7b593b..HEAD

```
7b593b..
```

□□□□

□Git□‘□□□□’□□□□□□□□□□□□□□□□. □□□□’□□□□'(Tracking Branches)□□□□□(push)□□□(pull)□,□□□□□□□(push)□□□(pull)□□□□□□□□□.

□□□□□□□□□□□□□□(pull)□□□□□,□□□□□□□□□□"git pull"□□□□; □□□□□□□‘□□□□'(Tracking Branches).

‘git clone‘□□□□□□□□□□□□□'master'□□□□□'origin/master'□‘□□□□’. □'origin/master'□□□□□(clone)□□□'master'□□.

□□□: origin□□□□□□□□□□□□□□.

□□□□□□'git branch'□□□□□'--track'□□, □□□□□□□'□□□□'.

```
git branch --track experimental origin/experimental
```

□□□□□□□□:

```
$ git pull experimental
```

□□□□□‘origin'□□(fetch)□□□□□□□□'origin/experimental'□□□□□(merge)□□□'experimental'□□.

□□□□□□□(push)□origin□, □□□□□□□'experimental'□□□□□□□□□origin□‘experimental'□□□,□□□□□□□(origin).

□□GIT GREP□□□□

□git grep □□□□Git□□□□□□□□□□□□□□. □□, □□□□□unix□□'grep'□□□□□□, □□'git grep'□□□□□□□□□(checkout)□□□□, □□□□□□.

□□, □□□ git.git□□□□□□□□□□□'xmmap'□□□□□, □□□□□□□□□□:

```
$ git grep xmmap
config.c:               contents = xmmap(NULL, contents_sz, PROT_READ,
diff.c:         s->data = xmmap(NULL, s->size, PROT_READ, MAP_PRIVATE, fd, 0);
git-compat-util.h:extern void *xmmap(void *start, size_t length, int prot, int fla
read-cache.c:   mmap = xmmap(NULL, mmap_size, PROT_READ | PROT_WRITE, MAP_PRIVATE,
refs.c: log_mapped = xmmap(NULL, mapsz, PROT_READ, MAP_PRIVATE, logfd, 0);
sha1_file.c:    map = xmmap(NULL, mapsz, PROT_READ, MAP_PRIVATE, fd, 0);
sha1_file.c:    idx_map = xmmap(NULL, idx_size, PROT_READ, MAP_PRIVATE, fd, 0);
sha1_file.c:                    win->base = xmmap(NULL, win->len,
sha1_file.c:                    map = xmmap(NULL, *size, PROT_READ, MAP_PRIVATE, f
sha1_file.c:            buf = xmmap(NULL, size, PROT_READ, MAP_PRIVATE, fd, 0);
wrapper.c:void *xmmap(void *start, size_t length,
```

□□□□□□□□, □□□□□'-n'□□:

```
$>git grep -n xmmap
config.c:1016:          contents = xmmap(NULL, contents_sz, PROT_READ,
diff.c:1833:            s->data = xmmap(NULL, s->size, PROT_READ, MAP_PRIVATE, fd,
git-compat-util.h:291:extern void *xmmap(void *start, size_t length, int prot, int
read-cache.c:1178:      mmap = xmmap(NULL, mmap_size, PROT_READ | PROT_WRITE, MAP_
refs.c:1345:    log_mapped = xmmap(NULL, mapsz, PROT_READ, MAP_PRIVATE, logfd, 0);
sha1_file.c:377:        map = xmmap(NULL, mapsz, PROT_READ, MAP_PRIVATE, fd, 0);
sha1_file.c:479:        idx_map = xmmap(NULL, idx_size, PROT_READ, MAP_PRIVATE, fd
sha1_file.c:780:                        win->base = xmmap(NULL, win->len,
sha1_file.c:1076:                       map = xmmap(NULL, *size, PROT_READ, MAP_PR
sha1_file.c:2393:               buf = xmmap(NULL, size, PROT_READ, MAP_PRIVATE, fd
wrapper.c:89:void *xmmap(void *start, size_t length,
```

□□□□□□□□□□□, □□□□□□'--name-only'□□:

```
$>git grep --name-only xmmap
config.c
diff.c
git-compat-util.h
read-cache.c
refs.c
sha1_file.c
wrapper.c
```

□□□□□'-c'□□,□□□□□□□□□□□□□□□□□(line matches):

```
$>git grep -c xmmap
config.c:1
diff.c:1
git-compat-util.h:1
read-cache.c:1
refs.c:1
sha1_file.c:5
wrapper.c:1
```

□□, □□□□□□□git□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□(tag reference):

```
$ git grep xmmap v1.5.0
v1.5.0:config.c:                contents = xmmap(NULL, st.st_size, PROT_READ,
v1.5.0:diff.c:          s->data = xmmap(NULL, s->size, PROT_READ, MAP_PRIVATE, fd,
v1.5.0:git-compat-util.h:static inline void *xmmap(void *start, size_t length,
v1.5.0:read-cache.c:                    cache_mmap = xmmap(NULL, cache_mmap_size,
v1.5.0:refs.c:  log_mapped = xmmap(NULL, st.st_size, PROT_READ, MAP_PRIVATE, logfd
v1.5.0:sha1_file.c:     map = xmmap(NULL, st.st_size, PROT_READ, MAP_PRIVATE, fd,
v1.5.0:sha1_file.c:     idx_map = xmmap(NULL, idx_size, PROT_READ, MAP_PRIVATE, fd
v1.5.0:sha1_file.c:                     win->base = xmmap(NULL, win->len,
v1.5.0:sha1_file.c:     map = xmmap(NULL, st.st_size, PROT_READ, MAP_PRIVATE, fd,
v1.5.0:sha1_file.c:             buf = xmmap(NULL, size, PROT_READ, MAP_PRIVATE, fd
```

□□□□□"1.5.0□"□□□□□□□□□□: □“1.5.0□"□, xmmap□□□wrapper.c□□□.

□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□'SORT_DIRENT'.

```
$ git grep -e '#define' --and -e SORT_DIRENT
builtin-fsck.c:#define SORT_DIRENT 0
builtin-fsck.c:#define SORT_DIRENT 1
```

□□□□□□□“□"(*both*)□□□□□□□□□□□□"□"(*either*)□□□□□□.

```
$ git grep --all-match -e '#define' -e SORT_DIRENT
builtin-fsck.c:#define REACHABLE 0x0001
builtin-fsck.c:#define SEEN      0x0002
builtin-fsck.c:#define ERROR_OBJECT 01
builtin-fsck.c:#define ERROR_REACHABLE 02
builtin-fsck.c:#define SORT_DIRENT 0
builtin-fsck.c:#define DIRENT_SORT_HINT(de) 0
builtin-fsck.c:#define SORT_DIRENT 1
builtin-fsck.c:#define DIRENT_SORT_HINT(de) ((de)->d_ino)
builtin-fsck.c:#define MAX_SHA1_ENTRIES (1024)
builtin-fsck.c: if (SORT_DIRENT)
```

□□□□□□□□□□□□□□(term)□□□□□□□(terms)□□□□□□.□□□□□□□□□□□□□‘PATH'□□'MAX'□□□□□:

```
$ git grep -e '#define' --and \( -e PATH -e MAX \)
abspath.c:#define MAXDEPTH 5
builtin-blame.c:#define MORE_THAN_ONE_PATH      (1u<<13)
builtin-blame.c:#define MAXSG 16
builtin-describe.c:#define MAX_TAGS     (FLAG_BITS - 1)
builtin-fetch-pack.c:#define MAX_IN_VAIN 256
builtin-fsck.c:#define MAX_SHA1_ENTRIES (1024)
...
```

□□□:□□□"□"□□□□□"□"□□□□□□□□□□.

GIT□□□□□ - □□, □□ □ □□

Git□□□□□□□□□□□□□□□□□□□. □□□□□□□□□□□□: □□□□□□□□□□□□□(commited); □□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□□□.

□□□□□□□□□□□(□□)

□□□□□□□□□□(work tree)□□□□□□□, □□□□□□□□□□□□□; □□□□□□□□□□, □□□□□□□□□□□□□□□(last committed state):

```
$ git reset --hard HEAD
```

□□□□□□□□□□□□□□□□□□□□□□(□□□□□□□□□□□□□□□□ untracked files). □□□□□□□□, □□□"git diff" □"git diff --cached"□□□□□□□□□□.

[illegible],[illegible]"hello.rb", [illegible]git checkout

```
$ git checkout -- hello.rb
```

[illegible]hello.rb[illegible]HEAD[illegible].

[illegible]:[illegible],[illegible].

[illegible]

[illegible](commit),[illegible], [illegible]:

1. [illegible](commit), [illegible]. [illegible].

2[illegible](old commit). [illegible],[illegible]; git[illegible],[illegible].

[illegible]

[illegible](revert)[illegible](commit)[illegible]; [illegible](commit)[illegible](reference)[illegible]git revert[illegible]; [illegible]:

```
$ git revert HEAD
```

[illegible](HEAD)[illegible], [illegible](new commit)[illegible].

[illegible], [illegible]“[illegible]”(next-to-last)[illegible]:

```
$ git revert HEAD^
```

[illegible],git[illegible],[illegible].[illegible](overlap),[illegible](conflicts),[illegible](merge)[illegible].

[illegible]: git revert [illegible](commit), [illegible](index)[illegible],[illegible]git commit[illegible](commit).

[illegible]

[illegible](commit), [illegible]; git commit [illegible]**--amend**[illegible](HEAD commit). [illegible],[illegible]
[illegible](commit message).

[illegible](older commit)[illegible], [illegible]. [illegible]git rebase[illegible], "git rebase -i"[illegible]. [illegible]rebase[illegible].

[illegible]GIT

[illegible]

[illegible], git[illegible].

[illegible], [illegible]git gc:

```
$ git gc
```

当仓库被压缩后, 请使用git gc会将对象收集起来以进行一次性处理.

悬空的对象

git fsck 命令有时会提示一些对象, 对于这些对象不必太担心. 在某些操作进行之后, 可能会留下一些被称为“悬空对象"(dangling objects).

```
$ git fsck
dangling commit 7281251ddd2a61e38657c827739c57015671a6b3
dangling commit 2706a059f258c6b245f298dc4ff2ccd30ec21a63
dangling commit 13472b7c4b80851a1bc551779171dcb03655e9b5
dangling blob 218761f9d90712d37a9c5e36f406f92202db07eb
dangling commit bf093535a34a4d35731aa2bd90fe6b176302f14f
dangling commit 8e4bec7f2ddaa268bef999853c25755452100f8e
dangling tree d50bb86186bf27b681d25af89d3b5b68382e4085
dangling tree b24c2473f1fd3d91352a624795be026d64c8841f
...
```

“悬空对象"(dangling objects)并不是问题, 它们通常是被重置或变基后的旧提交引起的. 你可以用垃圾回收命令对它们进行彻底清理.

建立一个公共仓库

假设你个人的仓库在目录 ~/proj. 我们先克隆一个新的“裸仓库“,并且创建一个标志文件告诉git-daemon这是个公共仓库.

```
$ git clone --bare ~/proj proj.git
$ touch proj.git/git-daemon-export-ok
```

上面的命令创建了一个proj.git目录, 这个目录里有一个“裸git仓库" -- 即只有'.git'目录里的内容,没有任何签出(checked out)的文件.

下一步就是你把这个 proj.git目录拷到你打算用来托管公共仓库的主机上. 你可以用scp, rsync或其它任何方式.

通过git协议导出git仓库

用git协议导出git仓库, 这是推荐的方法.

如果这台服务器上有管理员，TA们应该告诉你把仓库放在什么目录中, 并且“git:// URL”除仓库目录部分外是什么.

你现在要做的是启动git daemon; 它会监听在 9418端口. 默认情况下它会允许你访问所有的git目录(看目录中是否有git-daemon-export-ok文件). 如果以某些目录做为 git-daemon 的参数, 那么 git-daemon 会限制用户通过git协议只能访问这些目录.

你可以在inetd service模式下运行 git-daemon; 请看git daemon帮助里的相关内容.

通过http协议导出git仓库

git协议有不错的性能和可靠性, 但是如果主机上已经配好了一台web服务器,使用http协议(git over http)可能会更容易配置一些.

你需要把新建的"裸仓库"放到Web服务器的可访问目录里, 同时做一些调整,以便让web客户端获得它们所需的额外信息.

```
$ mv proj.git /home/you/public_html/proj.git
$ cd proj.git
$ git --bare update-server-info
$ chmod a+x hooks/post-update
```

([illegible]:git update-server-info & githooks.)

[illegible]proj.git[illegible]web URL, [illegible](clone)[illegible](pull) git[illegible]. [illegible]:

```
$ git clone http://yourserver.com/~you/proj.git
```

[illegible]

[illegible],[illegible], [illegible]:

[illegible]**SSH**[illegible]

[illegible]ssh[illegible]Git(Git Over SSH). [illegible]ssh[illegible], [illegible]“git[illegible]"[illegible]ssh[illegible], [illegible]ssh[illegible]. [illegible]
[illegible]. [illegible], [illegible]"[illegible]", [illegible]scp[illegible]:

```
$ git clone --bare /home/user/myrepo/.git /tmp/myrepo.git
$ scp -r /tmp/myrepo.git myserver.com:/opt/git/myrepo.git
```

[illegible] myserver.com[illegible]ssh[illegible]TA[illegible](clone)[illegible]:

```
$ git clone myserver.com:/opt/git/myrepo.git
```

[illegible]ssh[illegible](public key).

[illegible]1:[illegible]ssh[illegible],[illegible]ssh[illegible].

[illegible]2:git over ssh[illegible], git://[illegible].

[illegible]**Gitosis**[illegible]

[illegible],[illegible]Gitosis[illegible]. [illegible]gitosis[illegible], [illegible] authorized_keys [illegible](public key), [illegible]'git'[illegible]
(push)[illegible](pull)[illegible].

[illegible]Gitosis([illegible])

[illegible]1: github.com[illegible]([illegible])[illegible].

[illegible]: Gitosis[illegible]([illegible])

Chapter 5

□□□□

□□□□□□□

□□

```
git symbolic-ref HEAD refs/heads/newbranch
rm .git/index
git clean -fdx
<do work>
git add your files
git commit -m 'Initial commit'
```

gitcast:c9-empty-branch

□□□□□□

□□□□□□□□□□□□□□□□□

git filter-branch□□□□□□□□□□□□

□□□□□□□

□□□□□□□□□□□□□□□

git□□□□□□□□□□□□□□□□□□□□□, □git diff□□□□□□□□□□□. □□□□□□□□□□□□:

```
$ git diff
diff --cc file.txt
index 802992c,2b60207..0000000
--- a/file.txt
+++ b/file.txt
@@@ -1,1 -1,1 +1,5 @@@
++<<<<<<< HEAD:file.txt
 +Hello world
++=======
+ Goodbye
```

```
++>>>>>>> 77976da35a11db4580b80ae27e8d65caf5208086:file.txt
```

□□□□, □□□□□□□□□□, □□□□□□□□□□□□□□□□□: □□□□□□□□HEAD, □□□□□□□□tip; □□□□□□□□□□□□□□□□tip, □□□□□MERGE_HEAD.

□□□□□□, □□□□□□□□□□□□□□□. □□"□□□□(file stage)"□□□□□□□□□□□□□□□□:

```
$ git show :1:file.txt  # □□□□□□□□□□□□.
$ git show :2:file.txt  # HEAD□□□□.
$ git show :3:file.txt  # MERGE_HEAD□□□□.
```

□□□□git diff□□□□□□, □□□□□(work tree), □□2(stage 2)□□□3(stage 3)□□□□□□diff□□, □□□□□□□□□□□(□□□□, □□□□□□□□□□□□□2□□□□, □□□□□□□□□; □□□□3□ □□□□□).

□□□diff□□□□□file.txt□□□□, □□2□□□3□□□□. git□□□□□□□□□□'+'□□'-', □□□, □□□□□□□□□□: □□□□□□□□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□□. (□□git diff-files□□"COMBINED DIFF FORMAT"□□□□□□□□□□.)

□□□□□□□□□□□□□(□□□□□□□□□), diff□□□□□□□□□□:

```
$ git diff
diff --cc file.txt
index 802992c,2b60207..0000000
--- a/file.txt
+++ b/file.txt
@@@ -1,1 -1,1 +1,1 @@@
- Hello world
-Goodbye
++Goodbye world
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□"Hello world"□□□□□□□□□□"Goodbye", □□□□□□□□□□□□□□□□□□"Goodbye world".

□□□□diff□□□□□□□□□□□□□□□□□□□□□□□□:

```
$ git diff -1 file.txt      # □□□1□□□□
$ git diff --base file.txt          # □□□□
$ git diff -2 file.txt      # □□□2□□□□
$ git diff --ours file.txt          # □□□□
$ git diff -3 file.txt      # □□□3□□□□
$ git diff --theirs file.txt    # □□□□.
```

git log□gitk□□□□□□□□□□□□□□□□:

```
$ git log --merge
$ gitk --merge
```

□□□□□□□□□□HEAD□□□□MERGE_HEAD□□□□□□, □□□□□□(touch)□□□□□□□□□.

□□□□□□git mergetool, □□□□□□□□□□□emacs□kdiff3□□□□□.

□□□□□□□□□, □□□□□□:

```
$ git add file.txt
```

□□□□□□□□, git-diff(□□□)□□□□□□□□□□□, □□□□□□□□□□□□□□□□"□□"□□.

□□□□

[illegible], [illegible]git merge[illegible]. [illegible],

```
$ git merge scott/master rick/master tom/master
```

[illegible]:

```
$ git merge scott/master
$ git merge rick/master
$ git merge tom/master
```

[illegible]

[illegible]. [illegible], [illegible].

[illegible], [illegible]. [illegible]Makefile[illegible]. [illegible], [illegible]. [illegible]. [illegible]
[illegible].

[illegible], [illegible].

[illegible], [illegible]/path/to/B ([illegible], [illegible]URL). [illegible]master[illegible]dir-B[illegible].

[illegible]:

```
$ git remote add -f Bproject /path/to/B (1)
$ git merge -s ours --no-commit Bproject/master (2)
$ git read-tree --prefix=dir-B/ -u Bproject/master (3)
$ git commit -m "Merge B project as our subdirectory" (4)
$ git pull -s subtree Bproject master (5)
```

[illegible]. [illegible]([illegible]1.5.2)[illegible], [illegible].

[illegible], [illegible](submodule), [illegible]. [illegible].

[illegible]: submodule[illegible]Git[illegible].

[illegible], [illegible], [illegible].

(from Using Subtree Merge)

[illegible] - GIT BISECT

[illegible]'2.6.18'[illegible], [illegible](master)[illegible](crash)[illegible]. [illegible]: [illegible](brute-force regression)[illegible],[illegible](commit)[illegible]. [illegible]
linkgit:git-bisect1 [illegible]:

```
$ git bisect start
$ git bisect good v2.6.18
$ git bisect bad master
Bisecting: 3537 revisions left to test after this
[65934a9a028b88e83e2b0f8b36618fe503349f8e] BLOCK: Make USB storage depend on SCSI rather than selecting it [try #6]
```

[illegible]"git branch",[illegible]"no branch"([illegible]:[illegible]git bisect[illegible]). [illegible]commit):"69543", [illegible]"v2.6.18"[illegible]"master"[illegible]. [illegible]
[illegible],[illegible], [illegible](crash). [illegible], [illegible]:

```
$ git bisect bad
Bisecting: 1769 revisions left to test after this
[7eff82c8b1511017ae605f0c99ac275a7e21b867] i2c-core: Drop useless bitmaskings
```

□□git□□□□(checkout)□□□□□□□. □□□□□, □"git bisect good","git bisect bad"□□git□□□□□□□□□□□□□; □□□□□□□□□□□□□□□□□, □□□□git□□"□□□□(binary search)□□"□□"bad"□"good"□□□□□□□(commit or revison).

□□□□□(case)□, □□13□□□, □□□□□□□□□□(guilty commit). □□□□git show □□□□□□□□(commit), □□□□□□□□□□□□□□□TA. □□, □□:

```
$ git bisect reset
```

□□□□□□(□□git bisect start□□)□□□.

□□: git-bisect □□□□□□□□□□, □□□□□□; □□□□□□□□□, □□□□□□□□□□□□□□□□□.

□□: $ git bisect visualize

□□□□gitk, □□□□□□□"git bisect"□□□□□□□□□(commit). □□□□□□□□□□□□(commit), □□□□SHA□□, □□□□□□□□□□□:

```
$ git reset --hard fb47ddb2db...
```

□□□□□□, □□□□□□□□□□”bisect good"□□"bisect bad"; □□□□□□□, □□□□□□□□.

□□□: □□"git bisect start"□□□□□□, □□□□□□□□. □□□□□□"git bisect start"□□□□□□□□"bisect"□□□, □□□□□□□□□□"no branch"□□□.

□□□□□□□ - GIT BLAME

□□□□□□□□□□□□□□□□□□, □□git blame □□□□□□. □□□□'git blame [filename]', □□□□□□□□□□□□□□□□□□□□□:□□SHA□,□□□□□:

□□□: Git□□SHA1□□hash□□□□, □□□□,□□□□□□□□,□□□□SHA□□□SHA1. □□□□□□□□, □□□□SHA□□SHA1□□□.

```
$ git blame sha1_file.c
...
0fcfd160 (Linus Torvalds  2005-04-18 13:04:43 -0700    8)  */
0fcfd160 (Linus Torvalds  2005-04-18 13:04:43 -0700    9) #include "cache.h"
1f688557 (Junio C Hamano  2005-06-27 03:35:33 -0700   10) #include "delta.h"
a733cb60 (Linus Torvalds  2005-06-28 14:21:02 -0700   11) #include "pack.h"
8e440259 (Peter Eriksen   2006-04-02 14:44:09 +0200   12) #include "blob.h"
8e440259 (Peter Eriksen   2006-04-02 14:44:09 +0200   13) #include "commit.h"
8e440259 (Peter Eriksen   2006-04-02 14:44:09 +0200   14) #include "tag.h"
8e440259 (Peter Eriksen   2006-04-02 14:44:09 +0200   15) #include "tree.h"
f35a6d3b (Linus Torvalds  2007-04-09 21:20:29 -0700   16) #include "refs.h"
70f5d5d3 (Nicolas Pitre   2008-02-28 00:25:19 -0500   17) #include "pack-revindex.h"628522ec (Junio C Hamano              2007-12-29 02:05:47 -0800   18) #include "sha1-look
...
```

□□□□□□□□(reverted),□□□□(build)□□□; □□□□□□□□□□□□.

□□□□□"-L"□□□□□(blame)□□□□□□□□□:

```
$>git blame -L 160,+10 sha1_file.c
ace1534d (Junio C Hamano 2005-05-07 00:38:04 -0700       160)}
ace1534d (Junio C Hamano 2005-05-07 00:38:04 -0700       161)
0fcfd160 (Linus Torvalds 2005-04-18 13:04:43 -0700       162)/*
0fcfd160 (Linus Torvalds 2005-04-18 13:04:43 -0700       163) * NOTE! This returns a statically allocate
790296fd (Jim Meyering   2008-01-03 15:18:07 +0100       164) * careful about using it. Do an "xstrdup()
0fcfd160 (Linus Torvalds 2005-04-18 13:04:43 -0700       165) * filename.
```

```
ace1534d (Junio C Hamano 2005-05-07 00:38:04 -0700      166) *
ace1534d (Junio C Hamano 2005-05-07 00:38:04 -0700      167) * Also note that this returns the location
ace1534d (Junio C Hamano 2005-05-07 00:38:04 -0700      168) * SHA1 file can happen from any alternate
d19938ab (Junio C Hamano 2005-05-09 17:57:56 -0700      169) * DB_ENVIRONMENT environment variable if i
```

GIT□EMAIL

□□□□□□□□□

□□□□□□□□□□□, □□□□□□□□□□□□□□□□□(patch)□□□□□□:

□□, □□git format-patch; □□:

```
$ git format-patch origin
```

□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□origin/HEAD□□□□□□□.

□□□□□□□□□□□□□□□□Email□□□. □□□□□□□□□□□□□□□, □□□□□□□□git send-email□□□□□□□□□□□□□. □□□□□, □□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□.

□□□□□□□□□□

Git□□□□□□□□git am□□□(am□"apply mailbox"□□□)□□□□□□□Email□□□□□□□. □□□□□□□□□□□□□□□□□□□□□mailbox□□, □□□"patches.mbox", □□□□

```
$ git am -3 patches.mbox
```

Git□□□□□□□□□□□□; □□□□□□□, git□□□□□□□□□□□□□□□□□. ("-3"□□□□git□□□□□□; □□□□□□□□□□□□□□□□□□□□□, □□□□□"-3"□□.)

□□□□□□□□□□□□, □□□□□□□□□□□□, □□□□□

```
$ git am --resolved
```

□□git□□□□□□□□□, □□□□□□mailbox□□□□□□.

□□□□□□, git□□□□□□□□, □□□□□□□mailbox□□□□□□, □□□□□□□□□□□□□□□□□□□□□.

□□GIT

git config

□□□□□□□

```
$ git config --global core.editor emacs
```

□□□□

```
$ git config --global alias.last 'cat-file commit HEAD'

$ git last
tree c85fbd1996b8e7e5eda1288b56042c0cdb91836b
```

```
parent cdc9a0a28173b6ba4aca00eb34f5aabb39980735
author Scott Chacon <schacon@gmail.com> 1220473867 -0700
committer Scott Chacon <schacon@gmail.com> 1220473867 -0700

fixed a weird formatting problem

$ git cat-file commit HEAD
tree c85fbd1996b8e7e5eda1288b56042c0cdb91836b
parent cdc9a0a28173b6ba4aca00eb34f5aabb39980735
author Scott Chacon <schacon@gmail.com> 1220473867 -0700
committer Scott Chacon <schacon@gmail.com> 1220473867 -0700

fixed a weird formatting problem
```

[illegible]

[illegible]color.*[illegible]git config[illegible]

```
$ git config color.branch auto
$ git config color.diff auto
$ git config color.interactive auto
$ git config color.status auto
```

[illegible]color.ui[illegible]:

```
$ git config color.ui true
```

[illegible]

```
$ git config commit.template '/etc/git-commit-template'
```

[illegible]

```
$ git config format.pretty oneline
```

[illegible]

[illegible], [illegible], [illegible], [illegible], [illegible], http[illegible], diff, [illegible], [illegible]. [illegible]git, [illegible]git config[illegible].

GIT HOOKS

[illegible](hooks)[illegible]$GIT-DIR/hooks[illegible], [illegible](certain points)[illegible]git init[illegible], [illegible]hooks[illegible]; [illegible] [illegible]".sample"[illegible]

applypatch-msg

```
GIT_DIR/hooks/applypatch-msg
```

[illegible]git am[illegible](patch)[illegible](commit log message)[illegible]0[illegible]git am[illegible](apply the patch)[illegible]

[illegible]:[illegible]git am[illegible](commit)[illegible](message file)[illegible](commit)[illegible](inspect)[illegible]

□□□□□□□□□□(commit)□

□□□□□□applypatch-msg.sample□□□□□□□□□□□□commit-msg□□□□□□□□□□□□□□□□□□

pre-applypatch

GIT_DIR/hooks/pre-applypatch

□□□□□□□git am□□□□□□□□□□□□□□□□□□□□□□□□□□□□□(patch)□□□□□□□□□□□(commit)□□□□□□□□□□□□□□□□□□□□`0``□□□□□□□□□□□□□(patch)□□□□□□□□□□

□□(commit)□

□□□□□□pre-applypatch.sample□□□□□□□□□□□□□pre-commit□□□□□□□□□□□□□□□□□□□

post-applypatch

GIT_DIR/hooks/post-applypatch

□□□□□□□git am□□□□□□□□□□□□□□□□□□□□□□□□□□□□□(patch)□□□□□□□□□□□(commit)□□□□□□□□□□□

□□□□□□□□□□□□□□(notification)□□□□□□□□git-am□□□□□□□□

pre-commit

GIT_DIR/hooks/pre-commit

□□□□□□□ git commit □□□□, □□□□□□□□□□□□□□□\--no-verify □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□(commit)□□□□□□□□□□□□□□□□□0□□□□ git commit □□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□lint□□□□□

□□□□□pre-commit□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□git commit□□□□□□□□□□□□□□□□□□□□□□□□□□□□(commit message)□□□□□git-commit □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□GIT_EDITOR=:

□□□□□□□□□□ Rspec □□□□ Ruby □□□□□□□□□□□□□□□□□□□□□□□□□□□□□(commit)□

```
html_path = "spec_results.html"
`spec -f h:#{html_path} -f p spec` # run the spec. send progress to screen. save html results to html_path

# find out how many errors were found
html = open(html_path).read
examples = html.match(/(\d+) examples/)[0].to_i rescue 0
failures = html.match(/(\d+) failures/)[0].to_i rescue 0
pending = html.match(/(\d+) pending/)[0].to_i rescue 0

if failures.zero?
  puts "0 failures! #{examples} run, #{pending} pending"
else
  puts "\aDID NOT COMMIT YOUR FILES!"
  puts "View spec results at #{File.expand_path(html_path)}"
  puts
```

```
   puts "#{failures} failures! #{examples} run, #{pending} pending"
   exit 1
 end
```

prepare-commit-msg

```
GIT_DIR/hooks/prepare-commit-msg
```

[illegible]git commit[illegible](editor)[illegible]

It takes one to three parameters. The first is the name of the file

[illegible] * message[illegible]-m[illegible]-F[illegible] * template[illegible]-t[illegible]git config[illegible]commit.template[illegible] * merge[illegible](commit)[illegible](merge)[illegible].git/MERGE_MSG[illegible] * squash[illegible].git/SQUASH_MSG[illegible] * commit [illegible](commit)[illegible]SHA1[illegible]-c,-C[illegible]\--amend[illegible]

[illegible]0[illegible]git commit[illegible]

[illegible]\--no-verify[illegible]0[illegible](abort the commit)[illegible]pre-commit[illegible]

git[illegible]prepare-commit-msg.sample[illegible](a merge's commit message)[illegible]Conflicts:[illegible]

Chapter 6

Harry-Chen □□□□

commit-msg

GIT_DIR/hooks/commit-msg

[illegible]'git-commit'[illegible]\--no-verify[illegible]'git-commit'[illegible]

The hook is allowed to edit the message file in place, and can be used to normalize the message into some project standard format (if the project has one). It can also be used to refuse the commit after inspecting the message file.

[illegible](commit)[illegible]

The default 'commit-msg' hook, when enabled, detects duplicate "Signed-off-by" lines, and aborts the commit if one is found.

[illegible]'commit-msg'[illegible](Signed-off-by lines)[illegible](commit)[illegible]

post-commit

GIT_DIR/hooks/post-commit

[illegible]'git-commit'[illegible](commit)[illegible]

[illegible](notification)[illegible]'git-commit'[illegible]

pre-rebase

GIT_DIR/hooks/pre-rebase

[illegible]'git-base'[illegible]rebase[illegible]rebase([illegible]rebase)[illegible]

post-checkout

GIT_DIR/hooks/post-checkout

[illegible]'git-checkout'[illegible](worktree)[illegible]HEAD[illegible] ref[illegible]HEAD[illegible] ref[illegible]([illegible]1[illegible]0)[illegible]
[illegible]'git-checkout'[illegible]

[illegible]

post-merge

GIT_DIR/hooks/post-merge

This hook is invoked by 'git-merge', which happens when a 'git-pull' is done on a local repository. The hook takes a single parameter, a status flag specifying whether or not the merge being done was a squash merge. This hook cannot affect the outcome of 'git-merge' and is not executed, if the merge failed due to conflicts.

[illegible]

This hook can be used in conjunction with a corresponding pre-commit hook to save and restore any form of metadata associated with the working tree (eg: permissions/ownership, ACLS, etc). See contrib/hooks/setgitperms.perl for an example of how to do this.

pre-receive

GIT_DIR/hooks/pre-receive

This hook is invoked by 'git-receive-pack' on the remote repository, which happens when a 'git-push' is done on a local repository. Just before starting to update refs on the remote repository, the pre-receive hook is invoked. Its exit status determines the success or failure of the update.

[illegible]'git-push'[illegible]'git-receive-pack'[illegible]'git-receive-pack'[illegible] pre-receive [illegible]ref[illegible](exit status)[illegible]

This hook executes once for the receive operation. It takes no arguments, but for each ref to be updated it receives on standard input a line of the format:

[illegible](receive)[illegible](standard input)[illegible]ref[illegible]

SP SP LF

[illegible]SP[illegible]LF[illegible]

where <old-value> is the old object name stored in the ref, <new-value> is the new object name to be stored in the ref and <ref-name> is the full name of the ref. When creating a new ref, <old-value> is 40 0.

<old-value>[illegible]ref[illegible]<new-value>[illegible]ref[illegible]<ref-name>[illegible]ref[illegible]ref[illegible]<old-value>[illegible]40[illegible]0[illegible]

If the hook exits with non-zero status, none of the refs will be updated. If the hook exits with zero, updating of individual refs can still be prevented by the <> hook.

[illegible](ref)[illegible]<>[illegible]

Both standard output and standard error output are forwarded to 'git-send-pack' on the other end, so you can simply echo messages for the user.

[illegible](hook)[illegible](stdout & stderr)[illegible]'git-send-pack'[illegible](other end)[illegible](echo)[illegible]

If you wrote it in Ruby, you might get the args this way:

□□□□ruby,□□□□□□□□□□□□□□□□□□□□

```
rev_old, rev_new, ref = STDIN.read.split(" ")
```

Or in a bash script, something like this would work:

□bash□□□□□□□□□□□□□□□□

```
#!/bin/sh
# <oldrev> <newrev> <refname>
# update a blame tree
while read oldrev newrev ref
do
    echo "STARTING [$oldrev $newrev $ref]"
    for path in `git diff-tree -r $oldrev..$newrev | awk '{print $6}'`
    do
      echo "git update-ref refs/blametree/$ref/$path $newrev"
      `git update-ref refs/blametree/$ref/$path $newrev`
    done
done
```

update

```
GIT_DIR/hooks/update
```

□□□□□□□□□□□'git-push'□□□□□□□□□□□□□□□□□□□□'git-receive-pack'□□'git-receive-pack'□□□ update □□□□□□□□□□□□□ref□□□□□□□□□□□□□□□□□□□(exit status)□□□□update□□□□□

□□□□□□□(ref)□□□□□□□□□□□□□□□□□□□□:

- the name of the ref being updated, # □□□□ref□□
- the old object name stored in the ref, # ref □□□□□□□□
- and the new objectname to be stored in the ref. # ref □□□□□□□□

□□ update hook □□□□□□□□□□□□(ref)□□□□□□□□□□□□□□□□□□’git-receive-pack'□□□□□□□□□(ref)□

This hook can be used to prevent 'forced' update on certain refs by making sure that the object name is a commit object that is a descendant of the commit object named by the old object name. That is, to enforce a "fast forward only" policy.

□□□□□□□□□□□□□□□□□□□ refs□□□old object□new object□□□□□□□□□□□□□□□"fast forward only"□□□

It could also be used to log the old..new status. However, it does not know the entire set of branches, so it would end up firing one e-mail per ref when used naively, though. The <> hook is more suited to that.

□□□□□□□□(log)□□□□□□□□□□□□□□□□□□□ref□□□□□□□□□□□□□□□ref□□□□□email□□□><□□□□□□□□□□

□□□□□(mailing list)□□□□□□□□□□□□□□ update hook □□□□□(finer grained)□□□□□

□□(hook)□□□□□□□□□□□□□□□(stdout & stderr)□□□'git-send-pack'□□□□□(other end)□□□□□□□□□□□□(echo)□□□□

□□□□ update hook □□□□□hooks.allowunannotated□□□□□□□□□□□□□□(unannotated)□□□□□□□□□□□□□□□□

post-receive

`GIT_DIR/hooks/post-receive`

This hook is invoked by 'git-receive-pack' on the remote repository, which happens when a 'git-push' is done on a local repository. It executes on the remote repository once after all the refs have been updated.

[illegible]'git-push'[illegible]'git-receive-pack'[illegible](ref)[illegible]'git-receive-pack'[illegible]

This hook executes once for the receive operation. It takes no arguments, but gets the same information as the <> hook does on its standard input.

[illegible](receive)[illegible]<>[illegible](standard input)[illegible]

This hook does not affect the outcome of 'git-receive-pack', as it is called after the real work is done.

[illegible]'git-receive-pack'[illegible]

This supersedes the <> hook in that it gets both old and new values of all the refs in addition to their names.

[illegible] <>[illegible]ref[illegible]

Both standard output and standard error output are forwarded to 'git-send-pack' on the other end, so you can simply `echo` messages for the user.

[illegible](hook)[illegible](stdout & stderr)[illegible]'git-send-pack'[illegible](other end)[illegible](echo)[illegible]

The default 'post-receive' hook is empty, but there is a sample script `post-receive-email` provided in the `contrib/hooks` directory in git distribution, which implements sending commit emails.

[illegible]'post-receive'[illegible]git distribution`contrib/hooks` [illegible]`post-receive-email` [illegible]commit emails[illegible]

post-update

`GIT_DIR/hooks/post-update`

This hook is invoked by 'git-receive-pack' on the remote repository, which happens when a 'git-push' is done on a local repository. It executes on the remote repository once after all the refs have been updated.

[illegible]'git-push'[illegible]'git-receive-pack'[illegible](ref)[illegible]post-update [illegible]

It takes a variable number of parameters, each of which is the name of ref that was actually updated.

[illegible] ref[illegible]

This hook is meant primarily for notification, and cannot affect the outcome of 'git-receive-pack'.

[illegible](notification)[illegible]'git-receive-pack'[illegible]

The 'post-update' hook can tell what are the heads that were pushed, but it does not know what their original and updated values are, so it is a poor place to do log old..new. The <> hook does get both original and updated values of the refs. You might consider it instead if you need them.

'post-update'[illegible] heads [illegible]head[illegible]<>[illegible]ref([illegible]head)[illegible]

When enabled, the default 'post-update' hook runs 'git-update-server-info' to keep the information used by dumb transports (e.g., HTTP) up-to-date. If you are publishing a git repository that is accessible via HTTP, you should probably enable this hook.

□□□□□'post-update'□□□□□□□□□□□□‘git-update-server-info'□□□□□□□dumb□□(□http)□□□□□□□□□□□□git□□□□http□□□□□□□□□□□□□□□□□

Both standard output and standard error output are forwarded to 'git-send-pack' on the other end, so you can simply `echo` messages for the user.

□□(hook)□□□□□□□□□□□□□(stdout & stderr)□□□'git-send-pack'□□□□□□(other end)□□□□□□□□□□□□(echo)□□□□

pre-auto-gc

```
GIT_DIR/hooks/pre-auto-gc
```

□□□'git-gc --auto'□□□□□□□□□(hook)□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□'git-gc --auto'□□□□□□□□□

□□

Git Hooks * http://probablycorey.wordpress.com/2008/03/07/git-hooks-make-me-giddy/

□□□□□□□

□□□: □□□□□□□□□□□□Recovering Lost Commits Blog Post□□ Recovering Corrupted Blobs by Linus

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□git□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□git□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Let's go!

□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ BTW□□□□□□□□□□□□□□□□shell□□□□

```
mkdir recovery;cd recovery
git init
touch file
git add file
git commit -m "First commit"
echo "Hello World" > file
git add .
git commit -m "Greetings"
git branch cool_branch□
git checkout cool_branch
echo "What up world?" > cool_file
git add .
git commit -m "Now that was cool"
git checkout master
echo "What does that mean?" >> file
```

□□□□□□□□□

□□repo□□□□branch

```
$ git branch
cool_branch
* master
```

□□□□□□□□□□□□

```
$ git stash save "temp save"
Saved working directory and index state On master: temp save
HEAD is now at e3c9b6b Greetings
```

□□□□□□

```
$ git branch -D cool_branch
Deleted branch cool_branch (was 2e43cd5).

$ git branch
 * master
```

□`git fsck --lost-found`□□□□□□□□□□□□□□□□□□□

```
$git fsck --lost-found
  dangling commit 2e43cd56ee4fb08664cd843cd32836b54fbf594a
```

□git show□□□□□□□□□□□□□□□□□□□□□□□□□

```
git show 2e43cd56ee4fb08664cd843cd32836b54fbf594a

  commit 2e43cd56ee4fb08664cd843cd32836b54fbf594a
  Author: liuhui <liuhui998[#]gmail.com>
  Date:   Sat Oct 23 12:53:50 2010 +0800

  Now that was cool

  diff --git a/cool_file b/cool_file
  new file mode 100644
  index 0000000..79c2b89
  --- /dev/null
  +++ b/cool_file
  @@ -0,0 +1 @@
  +What up world?
```

□□

□□git rebase□□□□□

```
$git rebase 2e43cd56ee4fb08664cd843cd32836b54fbf594a
First, rewinding head to replay your work on top of it...
Fast-forwarded master to 2e43cd56ee4fb08664cd843cd32836b54fbf594a.
```

□□□□□git log□□□□□□□□□□□□□□:

```
$ git log

commit 2e43cd56ee4fb08664cd843cd32836b54fbf594a
Author: liuhui <liuhui998[#]gmail.com>
Date:   Sat Oct 23 12:53:50 2010 +0800
```

```
Now that was cool

commit e3c9b6b967e6e8c762b500202b146f514af2cb05
Author: liuhui <liuhui998[#]gmail.com>
Date:   Sat Oct 23 12:53:50 2010 +0800

Greetings

commit 5e90516a4a369be01b54323eb8b2660545051764
Author: liuhui <liuhui998[#]gmail.com>
Date:   Sat Oct 23 12:53:50 2010 +0800

First commit
```

□□□□□□□□□□□□□□□□□□□

```
liuhui@liuhui:~/work/test/git/recovery$ git branch
* master
```

□□git merge□□□□□

□□□□□□□□□□□□□

```
$ git reset --hard HEAD^
HEAD is now at e3c9b6b Greetings
```

□□□□□□□□□□□□

```
git fsck --lost-found
dangling commit 2e43cd56ee4fb08664cd843cd32836b54fbf594a
```

□□□□□□□□□□□□□□□□□

```
$ git merge 2e43cd56ee4fb08664cd843cd32836b54fbf594a
Updating e3c9b6b..2e43cd5
Fast-forward
cool_file |    1 +
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 cool_file
```

git stash□□□

□□□□git stash□□□□□□□□□□□□□□□□□□□□□□□□

□□repo□□□□□ $ git stash list stash@{0}: On master: temp save

□□□□□ $git stash clear liuhui@liuhui:~/work/test/git/recovery$ git stash list

□□git fsck --lost-found□□□ $git fsck --lost-found dangling commit 674c0618ca7d0c251902f0953987ff71860cb067

□git show□□□□□□□□□□

```
$git show 674c0618ca7d0c251902f0953987ff71860cb067

commit 674c0618ca7d0c251902f0953987ff71860cb067
```

```
Merge: e3c9b6b 2b2b41e
Author: liuhui <liuhui998[#]gmail.com>
Date:   Sat Oct 23 13:44:49 2010 +0800

    On master: temp save

diff --cc file
index 557db03,557db03..f2a8bf3
--- a/file
+++ b/file
@@@ -1,1 -1,1 +1,2 @@@
  Hello World
  ++What does that mean?
```

[illegible]

```
$ git merge 674c0618ca7d0c251902f0953987ff71860cb067
Merge made by recursive.
 file |    1 +
  1 files changed, 1 insertions(+), 0 deletions(-)
```

[illegible]

[illegible]The illustrated guide to recovering lost commits with Git,[illegible]

[illegible]Git Community Book[illegible]

[illegible]git fsck --lost-found[illegible]git[illegible]commit[illegible]dangling commit[illegible]dangling commit[illegible]git rebase[illegible]git merge[illegible]

[illegible]

[illegible], [illegible]. [illegible], [illegible]Linux[illegible]; [illegible]; [illegible].

[illegible], [illegible]. [illegible](checkout), [illegible]. [illegible], [illegible]API[illegible], [illegible].

Git[illegible](partial checkout), [illegible]([illegible])[illegible]. [illegible], [illegible], [illegible]git[illegible]. [illegible], [illegible].

[illegible], [illegible]. [illegible], [illegible], [illegible](vendor branch)[illegible]. [illegible]. [illegible], [illegible], [illegible], [illegible].

Git[illegible](submodule)[illegible]. [illegible](identity); [illegible]ID, [illegible]("superproject")[illegible]. [illegible]: [illegible]git[illegible], [illegible].

Git 1.5.3[illegible]git submodule[illegible]. Git 1.5.2[illegible]; [illegible].

[illegible], [illegible]4[illegible]:

```
$ mkdir ~/git
$ cd ~/git
$ for i in a b c d
do
    mkdir $i
```

```
        cd $i
        git init
        echo "module $i" > $i.txt
        git add $i.txt
        git commit -m "Initial commit, submodule $i"
        cd ..
    done
```

[illegible], [illegible]:

```
    $ mkdir super
    $ cd super
    $ git init
    $ for i in a b c d
    do
        git submodule add ~/git/$i $i
    done
```

[illegible]: [illegible], [illegible]!

[illegible]git-submodule[illegible]:

```
    $ ls -a
    .  ..  .git  .gitmodules  a  b  c  d
```

git-submodule add[illegible]:

- [illegible], [illegible]master[illegible].
- [illegible]gitmodules[illegible], [illegible], [illegible].
- [illegible]ID[illegible], [illegible].

[illegible]:

```
    $ git commit -m "Add submodules a, b, c and d."
```

[illegible]:

```
    $ cd ..
    $ git clone super cloned
    $ cd cloned
```

[illegible], [illegible]:

```
    $ ls -a a
    .  ..
    $ git submodule status
    -d266b9873ad50488163457f025db7cdd9683d88b a
    -e81d457da15309b4fef4249aba9b50187999670d b
    -c1536a972b9affea0f16e0680ba87332dc059146 c
    -d96249ff5d57de5de093e6baff9e0aafa5276a74 d
```

[illegible]: [illegible], [illegible]HEAD[illegible]. [illegible]git ls-remote ../git/a[illegible].

[illegible]. [illegible]git submodule init, [illegible]URL[illegible]git/config:

```
    $ git submodule init
```

[illegible]git-submodule update[illegible]:

```
$ git submodule update
$ cd a
$ ls -a
.  ..  .git  a.txt
```

git-submodule update□git-submodule add□□□□□□□□□□git-submodule update□□□□□□□□□□, □□□□□□□□tip. □□□□□□□□□□(tag): □□□□□, □□□□□□□□□□□□□□.

```
$ git branch
* (no branch)
master
```

□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□, □□□□, □□□□□□□□□, □□□□□□□□□□□□□□□□□□:

```
$ git checkout master
```

□□

```
$ git checkout -b fix-up
```

□□

```
$ echo "adding a line again" >> a.txt
$ git commit -a -m "Updated the submodule from within the superproject."
$ git push
$ cd ..
$ git diff
diff --git a/a b/a
index d266b98..261dfac 160000
--- a/a
+++ b/a
@@ -1 +1 @@
-Subproject commit d266b9873ad50488163457f025db7cdd9683d88b
+Subproject commit 261dfac35cb99d380eb966e102c1197139f7fa24
$ git add a
$ git commit -m "Updated submodule a."
$ git push
```

□□□□□□□□□□□□, □□□□git pull□□□□git submodule update.

□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□. □□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□:

```
$ cd ~/git/super/a
$ echo i added another line to this file >> a.txt
$ git commit -a -m "doing it wrong this time"
$ cd ..
$ git add a
$ git commit -m "Updated submodule a again."
$ git push
$ cd ~/git/cloned
$ git pull
$ git submodule update
error: pathspec '261dfac35cb99d380eb966e102c1197139f7fa24' did not match any file(s) known to git.
Did you forget to 'git add'?
Unable to checkout '261dfac35cb99d380eb966e102c1197139f7fa24' in submodule path 'a'
```

□□□□□□□□□□□□□□□□□, □□□□□□□□, □□□□□□□□□□□□□□□. □□□□□□□□, git□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.

```
$ cd ~/git/super/a
$ echo i added another line to this file >> a.txt
$ git commit -a -m "doing it wrong this time"
$ cd ..
$ git add a/
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       deleted:    a
#       new file:   a/a.txt
#
# Modified submodules:
#
# * a aa5c351...0000000 (1):
#   < Initial commit, submodule a
#
```

□□□□□□□□□□□, □□□□□□(reset)□□□□, □□□add□□□□□□□□□□□.

```
$ git reset HEAD A
$ git add a
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   a
#
# Modified submodules:
#
# * a aa5c351...8d3ba36 (1):
#   > doing it wrong this time
#
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.

□□□□□□□□□□□□□□□□□□□□□□□□□□□, □□git submodule update□□□□□. □□□□□□□□□□□□□□□□□□□□□.

```
$ cat a.txt
module a
$ echo line added from private2 >> a.txt
$ git commit -a -m "line added inside private2"
$ cd ..
$ git submodule update
Submodule path 'a': checked out 'd266b9873ad50488163457f025db7cdd9683d88b'
$ cd a
$ cat a.txt
module a
```

□□: □□□□□□□□□reflog□□□□□.

□□□□□□□□□□□, □□□□□□□□□□.

gitcast:c11-git-submodules

Chapter 7

Git□□□□

GIT □□ WINDOWS

(mSysGit)[http://code.google.com/p/msysgit/]

gitcast:c10-windows-git

□□GIT□□□□□□

Capistrano □ Git

GitHub Guide on Deploying with Cap

Git and Capistrano Screencast

□ SUBVERSION □□

□□□□□□□□□□□□GIT

□□□□□□□□□□□□□□□□□□□□Git, □□□□□□□□?

□Subversion□□

Git□□□□□□git-svn□□□, □□□□□(clone)□□, □□□□Subversion□□□□□□□□Git□□. GitHub□□□□□□□□□□□□.

```
$ git-svn clone http://my-project.googlecode.com/svn/trunk new-project
```

□□□□□□□□□□□□Subversion□□□□□□□□Git□□. □□□□□□□□□□□□, □□□□1□□□□, □□□□□□□□, □□□□□□□□□□□□.

□Perforce□□

[illegible]contrib/fast-import[illegible], [illegible]git-p4[illegible], [illegible]Perforce[illegible].

```
$ ~/git.git/contrib/fast-import/git-p4 clone //depot/project/main@all myproject
```

[illegible]

These are other SCMs that listed high on the Git Survey, should find import docs for them. !!TODO!!

- CVS
- Mercurial (hg)
- Bazaar-NG
- Darcs
- ClearCase

[illegible]GIT

Git[illegible].

[illegible]GUI

Git[illegible]Tcl/Tk[illegible]GUI[illegible]. Gitk[illegible], [illegible].

gitk

git gui[illegible], [illegible]add, remove[illegible]commit. [illegible], [illegible].

git gui

[illegible]

Mac[illegible]GitX and GitNub

Linux[illegible]Qt[illegible]QGit

GIT[illegible]

github [illegible](private)[illegible]

bitbucket [illegible]5[illegible]

repo.or.cz

GIT[illegible]

ContentDistribution

TicGit

GIT□□□□□

Ruby □ Git

grit

jgit + jruby

PHP □ Git

Python □ Git

pygit

Perl □ Git

perlgit

□□□:□□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□, □□□□□□□□□□:)

GIT □□□□

textmate

eclipse

netbeans

Chapter 8

□□□□

GIT□□□□□□

□□□□□□Git□□□□□□□.

□□□□□SHA□□□□gzip□□□□□, □□□□□□□□□, □□□□.

Git□□□□□ - □□□(loose object)□□□□(packed object).

□□□

□□□□□□□□□□. □□□□□□□□□□□□□□. □□□□□□□□□□□□.

□□□□SHA□ab04d884140f7b0cf8bbf86d6883869f16a46f65, □□□□□□□□□:

```
GIT_DIR/objects/ab/04d884140f7b0cf8bbf86d6883869f16a46f65
```

Git□SHA□□□□□□□□□□, □□□□□□□□□□□□□□. □□□□□□38□□.

□□□□Ruby□□□□□□□□□□□:

```ruby
def put_raw_object(content, type)
  size = content.length.to_s

  header = "#{type} #{size}\0" # type(space)size(null byte)
  store = header + content

  sha1 = Digest::SHA1.hexdigest(store)
  path = @git_dir + '/' + sha1[0...2] + '/' + sha1[2..40]

  if !File.exists?(path)
    content = Zlib::Deflate.deflate(store)

    FileUtils.mkdir_p(@directory+'/'+sha1[0...2])
    File.open(path, 'w') do |f|
      f.write content
    end
  end
  return sha1
end
```

[illegible]

[illegible](packfile). [illegible]Git[illegible], [illegible]. [illegible], Git[illegible], [illegible].

Git[illegible](packfile)[illegible]. [illegible], Git[illegible], [illegible]([illegible]: [illegible]).

[illegible], [illegible]. [illegible], [illegible] - [illegible]git gc[illegible]. [illegible], [illegible]. [illegible], [illegible], [illegible](git unpack-objects)[illegible](git repack).

Git[illegible]. [illegible], [illegible]SHA[illegible].

[illegible]"[illegible]"(Packfile)[illegible].

[illegible]GIT[illegible]

[illegible]cat-file[illegible]. [illegible]SHA[illegible], [illegible]40[illegible]:

```
$ git-cat-file -t 54196cc2
commit
$ git-cat-file commit 54196cc2
tree 92b8b694ffb1675e5975148e1121810081dbdffe
author J. Bruce Fields <bfields@puzzle.fieldses.org> 1143414668 -0500
committer J. Bruce Fields <bfields@puzzle.fieldses.org> 1143414668 -0500

initial commit
```

[illegible](tree)[illegible](blob)[illegible], [illegible]. [illegible], [illegible], [illegible]. [illegible]ls-tree[illegible]:

```
$ git ls-tree 92b8b694
100644 blob 3b18e512dba79e4c8300dd08aeb37f8e728b8dad    file.txt
```

[illegible]. SHA[illegible]([illegible]: [illegible]).

```
$ git cat-file -t 3b18e512
blob
```

[illegible]"[illegible]"(blob)[illegible], [illegible]cat-file[illegible]:

```
$ git cat-file blob 3b18e512
hello world
```

[illegible]. [illegible].

[illegible]SHA1[illegible]git[illegible]:

```
$ find .git/objects/
.git/objects/
.git/objects/pack
.git/objects/info
.git/objects/3b
.git/objects/3b/18e512dba79e4c8300dd08aeb37f8e728b8dad
.git/objects/92
.git/objects/92/b8b694ffb1675e5975148e1121810081dbdffe
.git/objects/54
.git/objects/54/196cc2703dc165cbd373a65a4dcf22d50ae7f7
.git/objects/a0
```

```
.git/objects/a0/423896973644771497bdc03eb99d5281615b51
.git/objects/d0
.git/objects/d0/492b368b66bdabf2ac1fd8c92b39d3db916e59
.git/objects/c4
.git/objects/c4/d59f390b9cfd4318117afde11d601c1085f241
```

□□□□□□□□□□□□□□□□□□□□□□□□□□. □□□□□□(blob), □(tree), □□(commit)□□□□(tag).

□□□□□□□HEAD□□, □□□□□.git/HEAD□□□:

```
$ cat .git/HEAD
ref: refs/heads/master
```

□□□□, □□□□□□□□□□□□□□□□□□□□□□□. Git□□□□.git□□□□□□□□□□□□(□□: □refs/heads□□□□□, □□□□□□□□□□□). □□□□□□□□□□□□□□SHA1□, □□□□□cat-file□□□□□□□□□
(□□: □□□□□□□□□□□):

```
$ cat .git/refs/heads/master
c4d59f390b9cfd4318117afde11d601c1085f241
$ git cat-file -t c4d59f39
commit
$ git cat-file commit c4d59f39
tree d0492b368b66bdabf2ac1fd8c92b39d3db916e59
parent 54196cc2703dc165cbd373a65a4dcf22d50ae7f7
author J. Bruce Fields <bfields@puzzle.fieldses.org> 1143418702 -0500
committer J. Bruce Fields <bfields@puzzle.fieldses.org> 1143418702 -0500

add emphasis
```

□□□□□□□□□□□□□□□□□□□□:

```
$ git ls-tree d0492b36
100644 blob a0423896973644771497bdc03eb99d5281615b51    file.txt
$ git cat-file blob a0423896
hello world!
```

□□□□□□□□□□□□□□:

```
$ git-cat-file commit 54196cc2
tree 92b8b694ffb1675e5975148e1121810081dbdffe
author J. Bruce Fields <bfields@puzzle.fieldses.org> 1143414668 -0500
committer J. Bruce Fields <bfields@puzzle.fieldses.org> 1143414668 -0500
```

GIT□□

□□(branch), □□□□□□□(remote-tracking branch)□□□□(tag)□□□□□□□□□. □□□□□□□"refs"□□, □□□□□□□□□. □□□□□, □□□□□□□□□□□□□□□□□□□□□□:

```
- □□"test"□"refs/heads/test"□□□.
- □□"v2.6.18"□"refs/tags/v2.6.18"□□□.
- "origin/master"□"refs/remotes/origin/master"□□□.
```

□□□□□□□□□□□□□□, □□□□□□□□□□□□□□, □□□□□□□□□□□□□.

(□□□□□□□□□□□□□□□□□□□.git/refs□□□. □□, □□□□□□, □□□□□□□□□□□□□□□□, □□git pack-refs).

□□□□□□□□□, □□□□□□□□□□□□□□HEAD. □□, "origin"□□□"origin"□□HEAD□□□□□□□.

[illegible]Git[illegible], [illegible], [illegible]git rev-parse[illegible]"SPECIFYING REVISIONS".

[illegible]

[illegible]"master"[illegible](reachable)[illegible].

[illegible]git show-ref[illegible]:

```
$ git show-ref --heads
bf62196b5e363d73353a9dcf094c59595f3153b7 refs/heads/core-tutorial
db768d5504c1bb46f63ee9d6e1772bd047e05bf9 refs/heads/maint
a07157ac624b2524a059a3414e99f6f44bebc1e7 refs/heads/master
24dbc180ea14dc1aebe09f14c8ecf32010690627 refs/heads/tutorial-2
1e87486ae06626c2f31eaa63d26fc0fd646c8af2 refs/heads/tutorial-fixes
```

[illegible]cut[illegible]grep[illegible]"[illegible]-[illegible]"(branch-head)[illegible], [illegible]"master":

```
$ git show-ref --heads | cut -d' ' -f2 | grep -v '^refs/heads/master'
refs/heads/core-tutorial
refs/heads/maint
refs/heads/tutorial-2
refs/heads/tutorial-fixes
```

[illegible]master[illegible]:

```
$ gitk master --not $( git show-ref --heads | cut -d' ' -f2 |
            grep -v '^refs/heads/master' )
```

[illegible]; [illegible]:

```
$ gitk $( git show-ref --heads ) --not  $( git show-ref --tags )
```

(git rev-parse[illegible]"--not"[illegible]"[illegible]"[illegible].)

(!!update-ref!!)

GIT[illegible]

[illegible](index)[illegible]([illegible].git/index), [illegible]SHA1[illegible]git ls-files[illegible]:

```
$ git ls-files --stage
100644 63c918c667fa005ff12ad89437f2fdc80926e21c 0   .gitignore
100644 5529b198e8d14decbe4ad99db3f7fb632de0439d 0   .mailmap
100644 6ff87c4664981e4397625791c8ea3bbb5f2279a3 0   COPYING
100644 a37b2152bd26be2c2289e1f57a292534a51a93c7 0   Documentation/.gitignore
100644 fbefe9a45b00a54b58d94d06eca48b03d40a50e0 0   Documentation/Makefile
...
100644 2511aef8d89ab52be5ec6a5e46236b4b6bcd07ea 0   xdiff/xtypes.h
100644 2ade97b2574a9f77e7ae4002a4e07a6a38e46d07 0   xdiff/xutils.c
100644 d5de8292e05e7c36c4b68857c1cf9855e3d2f70a 0   xdiff/xutils.h
```

[illegible], [illegible], [illegible]"[illegible](current directory cache)"[illegible]"[illegible](cache)". [illegible]:

1. [illegible]([illegible])[illegible].

[illegible], [illegible]git commit[illegible], [illegible](object database)[illegible], [illegible]. ([illegible]: [illegible]"[illegible]Git[illegible]"[illegible], [illegible].)

2. [illegible].

   [illegible]([illegible])[illegible]. [illegible], [illegible], [illegible], [illegible]Git[illegible].

3. [illegible], [illegible], [illegible].

   [illegible], [illegible]([illegible]"stages"). [illegible]git ls-files[illegible]stage[illegible]. [illegible], [illegible], [illegible]0.

[illegible](temporary staging area), [illegible].

[illegible]

[illegible](packfile)[illegible](packfile index)[illegible].

[illegible]

[illegible], [illegible], [illegible].

[illegible]. [illegible]1[illegible]Git 1.6[illegible], [illegible]2[illegible]Git 1.6[illegible]. [illegible]2[illegible]Git 1.5.2[illegible]Git[illegible], [illegible](backport)[illegible]1.4.4.5[illegible].

[illegible]2[illegible]CRC[illegible], [illegible], [illegible](from pack to pack)[illegible]. [illegible]2[illegible]4G[illegible].

[illegible], fanout([illegible])[illegible]SHA[illegible]. offset/sha1[illegible]SHA1[illegible]([illegible]), fanout[illegible]offset/sha1[illegible]([illegible]
[illegible]Hash[illegible], [illegible]8[illegible]).

[illegible]1[illegible], offset([illegible])[illegible]SHA[illegible]. [illegible]2[illegible], SHA[illegible], CRC[illegible]offset[illegible]. [illegible]CRC[illegible].

[illegible], [illegible](extract)[illegible], [illegible]. [illegible]. [illegible]"[illegible]"(upload-pack)[illegible]"[illegible]"(receive-pack)[illegible]([illegible]: [illegible]push
[illegible]fetch[illegible])[illegible](packfile format)[illegible], [illegible] - [illegible].

[illegible]

[illegible]. [illegible](header)[illegible]([illegible]header[illegible]body), [illegible](trailer). [illegible]4[illegible]'PACK', [illegible]. [illegible]4[illegible]
[illegible], [illegible]4[illegible](entry)[illegible]. [illegible]Ruby[illegible]:

```ruby
def read_pack_header
  sig = @session.recv(4)
  ver = @session.recv(4).unpack("N")[0]
  entries = @session.recv(4).unpack("N")[0]
  [sig, ver, entries]
end
```

[illegible]SHA[illegible], [illegible]. [illegible]([illegible])SHA[illegible]SHA1[illegible](20[illegible])([illegible]: [illegible]SHA1[illegible]).

[illegible](object header)[illegible]1[illegible], [illegible]. [illegible]7[illegible], [illegible]1[illegible]. [illegible]1[illegible]'1', [illegible]1[illegible]([illegible]: [illegible]

□), □□□□□□□□□□□□. □□□□□□□□3□□□□□□□□□□□, □□□□□□□□□□.

(3□□□□□□□□□□8□□. □□□□□□□□, 0(000)□'□□□□', 5(101)□□□□□□□□.)

□□□□□□□□□□□□□□□□. □1□□□□□3□□□□□□□□□□□(commit), □□□4□□□2□□□7□□□□□□□144, □□□□□□□□□□□144□□.

□

□□□□□□□□, □□□□□□□□'□□'□□□□□□□□□□□□, □□□*□□□□*□□□. □□, □□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.

□□□delta□□, □□□□□□□zlib□□□□□□□□. □□□□□delta□□, □□□□□□□□□□□□□□□□(base object)□□□□□□□□□delta(□□)□□. □□□□20□□□□□ref-delta, □□□□□SHA□□□20□□□. ofs-delta□□□□□□□□□□□□□□□□. □□□□□, □□□□□□□□□□□:

- delta□□□□□□□□□□□□□□□□;
- delta□□□□□□□□□□□□□□(□tree□tree, blob□blob, □□).

□□□□GIT

□□□□□□□□□□□□□□□□Git, □□□□□□□□□□□□□□□□□□blob(□), tree(□)□□commit(□□)□□. □□□□□□□□□□□Git□□□□□, □□□□□□□□□□□□.

□□blob□□

□□□Git□□□□□□□blob□□□□□□□□SHA□□□□□□, □□git hash-object□□□□. □□□□□□□□□□□□□□blob, □□'-w'□□□□□□□□□□□□□('-w'□□□□Git□□□blob, □□□□□□□SHA□).

```
$ git hash-object -w myfile.txt
6ff87c4664981e4397625791c8ea3bbb5f2279a3

$ git hash-object -w myfile2.txt
3bb0e8592a41ae3185ee32266c860714980dbed7
```

□□□□□□□□□□□□□□blob□SHA□.

□□tree□□

□□□□□□□□□□□□□□□□□□□□, □□git ls-tree□□□□□□□□,git mktree□□□□□□□□□□tree□□. □□, □□□□□□□□□□□□□'/tmp/tree.txt'□:

```
100644 blob 6ff87c4664981e4397625791c8ea3bbb5f2279a3    file1
100644 blob 3bb0e8592a41ae3185ee32266c860714980dbed7    file2
```

□□□□□□□□□□□□□□git mktree□, Git□□□□□□□tree□□, □□□□□□□□□□(object database)□, □□□□tree□□□SHA□.

```
$ cat /tmp/tree.txt | git mk-tree
f66a66ab6a7bfe86d52a66516ace212efa00fe1f
```

□□, □□□□□□□□□□tree□□□□□□tree□□□□, □□□□. □□□□□□□□□□□□□□□□□(□□□□□□□□□□□tree□□), □□□□□□□□□(/tmp/newtree.txt), □□□□tree□□□SHA□□□:

```
100644 blob 6ff87c4664981e4397625791c8ea3bbb5f2279a3    file1-copy
040000 tree f66a66ab6a7bfe86d52a66516ace212efa00fe1f    our_files
```

□□□□□□git mk-tree:

```
$ cat /tmp/newtree.txt | git mk-tree
5bac6559179bd543a024d6d187692343e2d8ae83
```

□□□□□□□□□□□□□□□□□:

```
.
|-- file1-copy
`-- our_files
    |-- file1
    `-- file2

1 directory, 3 files
```

□□□□□□□□□□□□□□□. □□, □□□□SHA□□□□□(5bac6559).

□□□□□

□□□□□□□□□□□□□□□□□□□□□□. □□□□□□□, □□□□□□□□□□□□□□□□□□□□, □□□5bac6559□□□□□□□□. (□□GIT_INDEX_FILE□□□□□□□□□□□□□□)

□□, □git read-tree□□□□□□□□□□□□□□, □□□□□□□□□□□□; □□□git write-tree□□□□□□□□□□□□□□:

```
$ export GIT_INDEX_FILE=/tmp/index
$ git read-tree --prefix=copy1/  5bac6559
$ git read-tree --prefix=copy2/  5bac6559
$ git write-tree
bb2fa6de7625322322382215d9ea78cfe76508c1

$>git ls-tree bb2fa
040000 tree 5bac6559179bd543a024d6d187692343e2d8ae83    copy1
040000 tree 5bac6559179bd543a024d6d187692343e2d8ae83    copy2
```

□□□□□□□□, □□□□□□□□□□□□□□□□□. □□□□□□□□□□□□□□□□□□ - □git read-tree□□□□□□.

□□**commit**□□

□□□□□□□□□□SHA□, □□□□□git commit-tree□□□□□□□□□□commit□□. □□□commit□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□:

```
GIT_AUTHOR_NAME
GIT_AUTHOR_EMAIL
GIT_AUTHOR_DATE
GIT_COMMITTER_NAME
GIT_COMMITTER_EMAIL
GIT_COMMITTER_DATE
```

□□□□□□□□□□□□□□□□□□□□□□□□□□git commit-tree, □□□□□□commit□□.

```
$ git commit-tree bb2fa < /tmp/message
a5f85ba5875917319471dfd98dfc636c1dc65650
```

□□□□□□□□□□□□□commit□□, □□□□□'-p'□□□□□□□□□commit□□. □□□, □□□□SHA□□□STDOUT□□.

□□□□□□□

□□□□□□□□□commit□□□SHA□, □□□□, □□□□□□□□□□□□□. □□□□□□□□□'master'□□□□□, □□□□□□□□□□□□, □□□git update-ref□□□□□□□:

```
$ git update-ref refs/heads/master a5f85ba5875917319471dfd98dfc636c1dc65650
```

□□□□

□□□□□□□□: Git□□□□□□□□□□□□□□□□.

□□**HTTP**□□□□

□□http□□□url□□□git□□□□, □□□□□□□□□□(dumber)□□□.

□□http□□, □□□□□□□(logic)□□□□□□□□. □□□□□□□□□□□□, □□□□git□□□□□□□□□□□web□□□□.

□□□□□http□□, □□□□□□□□□□□, □□□□□□□□git update-server-info. □□web□□□□□□□□□□□□□□□□□□□□, □git update-server-info□□□□□□□□□□(packfile)□□□(refs)□□□□□“objects/info/packs","info/refs"□□□□□□□. □ git update-server-info □□□,"objects/info/packs"□□□□□□□□□□□□:

```
P pack-ce2bd34abc3d8ebc5922dc81b2e1f30bf17c10cc.pack
P pack-7ad5f5d05f5e20025898c95296fe4b9c861246d8.pack
```

□□□□□http□□□□□□□□□□□□□□□□□(loose file), git□□□□□□□□□□□(packfiles). "info/refs" □□□□□□□□□□□□□:

```
184063c9b594f8968d61a686b2f6052779551613    refs/heads/development
32aae7aef7a412d62192f710f2130302997ec883    refs/heads/master
```

□□□□□□□□□□□(fetch)□□□, git□□□□□□□(refs)□□□□□□□□□□□□□(commit objects), □□□□□□□□□□□□□□□□□□□□.

□□, □□□□□(fetch)□□□□□"master"□□; git□□□□□□□"master"□□□32aae7ae, □□□□□"master"□□□□□ab04d88. □□□□□, □□□□32aae7ae□□□□.

□□□□□□□□□□□□(http□□□):

```
CONNECT http://myserver.com
GET /git/myproject.git/objects/32/aae7aef7a412d62192f710f2130302997ec883 - 200
```

□□□□□□□□□□□□□□□:

```
tree aa176fb83a47d00386be237b450fb9dfb5be251a
parent bd71cad2d597d0f1827d4a3f67bb96a646f02889
author Scott Chacon <schacon@gmail.com> 1220463037 -0700
committer Scott Chacon <schacon@gmail.com> 1220463037 -0700

added chapters on private repo setup, scm migration, raw git
```

□□□□□□□□□□□□□□□□(tree)aa176fb8: □□□:32aae7ae□□□□(commit object)□□□□□□(tree)□:aa176fb8.

```
GET /git/myproject.git/objects/aa/176fb83a47d00386be237b450fb9dfb5be251a - 200
```

□□□□□□□□□□□(tree)□□:

```
100644 blob 6ff87c4664981e4397625791c8ea3bbb5f2279a3    COPYING
100644 blob 97b51a6d3685b093cfb345c9e79516e5099a13fb    README
100644 blob 9d1b23b8660817e4a74006f15fae86e2a508c573    Rakefile
```

□□□, □□□(tree)□□3□□□(blob). □□, □□□□□□□□□□:

```
GET /git/myproject.git/objects/6f/f87c4664981e4397625791c8ea3bbb5f2279a3 - 200
GET /git/myproject.git/objects/97/b51a6d3685b093cfb345c9e79516e5099a13fb - 200
GET /git/myproject.git/objects/9d/1b23b8660817e4a74006f15fae86e2a508c573 - 200
```

□□http□□□□□□□□□curl□□□□, □□□□□□□□□□□□□□□□□□□. Git□□□□□□□(commit)□□□□□□□(tree)□, □□□□□□□□□□(commit)□□□□(next parent).

```
GET /git/myproject.git/objects/bd/71cad2d597d0f1827d4a3f67bb96a646f02889 - 200
```

□□□□□□(parent commit object)□□□□□□□□:

```
tree b4cc00cf8546edd4fcf29defc3aec14de53e6cf8
parent ab04d884140f7b0cf8bbf86d6883869f16a46f65
author Scott Chacon <schacon@gmail.com> 1220421161 -0700
committer Scott Chacon <schacon@gmail.com> 1220421161 -0700

added chapters on the packfile and how git stores objects
```

□□□□□□□□`ab04d88`□□□□□□(commit)□□□□, □`ab04d88`(commit)□□□□□□□"master"□□. □□□□□□□□□□□□(tree)`b4cc00c`□□□□, □□□□□□□□□□(commit)□□□□□. □□□□□□, □□□□□□'--recover'□□, □□git□□□□□□□□□□□□□□□□. □□□□□□: git http-fetch □□□□□□:

□□□□□□□□□(loose object)□□□□□, git□□□□□□□□□(packfile indexes), □□□□□□□□□sha□□□□□□□□□□□□□□□(packfile).

□□□□□git□□□□□□□□□□"post-receive"□□(hook), □□□□(hook)□□□□□□□□□'git update-server-info; □□□□□□□□□□□□□□□.

□□ Upload Pack □□□□

□□□□□□□□□□, □□□□□□□(fetching objects)□□□□□□. □□□□□□ssh□□□□git□□(git:// □□□□9418□□□□□), □□□□□□□□□□□□□socket□□□□□□□□□□□□git:fetch-pack □□, □□□□□□(fork)□ linkgit:git update-pack□□□□.

□□□□□□□□□□□□□□(ref)□□□□□SHA□□, □□□□□□□□□□□□□□□SHA□□□□□□.

□□, □□□□□□□□□□□□□□□□□□□(packfile), □□□□□□□□□.

□□□□□□□□□.

□□□□□□□□□□□□(request header). □□□□□□□:

```
$ git clone git://myserver.com/project.git
```

□□□□□□□□□□□□□□:

```
0032git-upload-pack /project.git\000host=myserver.com\000
```

□□□□□□□4□□□□□□□□16□□□□□(hex length) (□□□□4□□□,□□□□□□□). □□□□□□□□□□□, □□□□□□null□□(#body00)□□□□□. □□□□□□□null□□(\000)□□□.

□□□□□□□□□□□□□□□□"git-upload-pack"□□□□□.

```
$ git-upload-pack /path/to/repos/project.git
```

□□□□□□□□□□□□□□□:

```
007c74730d410fcb6603ace96f1dc55ea6196122532d HEAD\000multi_ack thin-pack side-band side-band-64k ofs-delta shallow no-progress
003e7d1665144a3a975c05f1f43902ddaf084e784dbe refs/heads/debug
003d5a3f6be755bbb7deae50065988cbfa1ffa9ab68a refs/heads/dist
003e7e47fe2bd8d01d481f44d7af0531bd93d3b21c01 refs/heads/local
003f74730d410fcb6603ace96f1dc55ea6196122532d refs/heads/master
```

```
0000
```

□□□□□□□4□□□□□□□□□□(□16□□□□). □□(section)□□□□□“0000”□□□□□□.

□□□□□□□□□□□□□□□□□□□. □□□□□□□□□□□□□□□□:

```
0054want 74730d410fcb6603ace96f1dc55ea6196122532d multi_ack side-band-64k ofs-delta
```

p 0032want 7d1665144a3a975c05f1f43902ddaf084e784dbe 0032want 5a3f6be755bbb7deae50065988cbfa1ffa9ab68a 0032want 7e47fe2bd8d01d481f44d7af0531bd93d3b21c01 0032want 74730d410fcb6603ace96f1dc55ea6196122532d 00000009done

□□□□□□□□□□□□□□□□"git-upload-pack"□□, □□□□□□□(streams out)□□□□□(final response):

```
"0008NAK\n"
"0023\002Counting objects: 2797, done.\n"
"002b\002Compressing objects:   0% (1/1177)   \r"
"002c\002Compressing objects:   1% (12/1177)   \r"
"002c\002Compressing objects:   2% (24/1177)   \r"
"002c\002Compressing objects:   3% (36/1177)   \r"
"002c\002Compressing objects:   4% (48/1177)   \r"
"002c\002Compressing objects:   5% (59/1177)   \r"
"002c\002Compressing objects:   6% (71/1177)   \r"
"0053\002Compressing objects:   7% (83/1177)   \rCompressing objects:   8% (95/1177)   \r"
...
"005b\002Compressing objects: 100% (1177/1177)   \rCompressing objects: 100% (1177/1177), done.\n"
"2004\001PACK\000\000\000\002\000\000\n\355\225\017x\234\235\216K\n\302"...
"2005\001\360\204{\225\376\330\345]z2673"...
...
"0037\002Total 2797 (delta 1799), reused 2360 (delta 1529)\n"
...
"<\276\255L\273s\005\001w0006\001[0000"
```

□□□□□"□□□□"(packfile)□□□, □□□□□□□□□□□□(packfile)□□□.

□□□□

□□git□ssh□□□□□□(pushing data)□□□□, □□□□□. □□□□, □□□□□□□"receive-pack"□□□, □□□□□□□□□□, □□□□□□□□□□□□"□"□SHA□□(all ref head shas). □□□
□□□□□, □□□□□□□□□□□□□□□□, □□□□□□□□□(packfile)□□□□□□. □□□□□□□□□□□□□□□□□□□□□□□□, □□□□□□□(□□□□□□□□□□□).

□□□□□□□□□□□□, □□□□□git push □□□□:git sendpack□□, □□□□□□"ssh□□□□"□□"git□□□"□□□:linkgit:git-receive-pack □□□□□□□□□.

□□□

□□□□Git□□□□□□□□□□□□□□□□□□□□□□(terms)□□□Git Glossary□

alternate object database

> Via the alternates mechanism, a repository can inherit part of its object database from another object database, which is called "alternate".

bare repository

> A bare repository is normally an appropriately named directory with a `.git` suffix that does not have a locally checked-out copy of any of the files under revision control. That is, all of the `git` administrative and control files that would normally be present in

the hidden `.git` sub-directory are directly present in the `repository.git` directory instead, and no other files are present and checked out. Usually publishers of public repositories make bare repositories available.

□□□

```
A bare repository is normally an appropriately
named directory with a `.git` suffix that does not
have a locally checked-out copy of any of the files under
revision control. That is, all of the `git`
administrative and control files that would normally be present in the
hidden `.git` sub-directory are directly present in the
`repository.git` directory instead,
and no other files are present and checked out. Usually publishers of
public repositories make bare repositories available.
```

blob object□□□□□□□

□□□□□□□□□□□□□□□□□□□□□

branch

A "branch" is an active line of development. The most recent commit on a branch is referred to as the tip of that branch. The tip of the branch is referenced by a branch head, which moves forward as additional development is done on the branch. A single git repository can track an arbitrary number of branches, but your working tree is associated with just one of them (the "current" or "checked out" branch), and HEAD points to that branch.

□□

```
□□“□□”□□□□□□□□(active line)□□□□
```

cache□□□□

□□(index)□□□(obsolete).

chain□□□□

□□□□□□□□□□□□□□□□□□□□□□□(reference to its successor)□□□□□□□□(commit)□□□□□□□□□□□□□

changeset□□□□□

BitKeeper/cvsps □□□□□(commit)□□□□□□ git □□□□□(states)□□□□□□□□□□□□□□ git □□□□□□□□

checkout□□□□

```
□□□□□□(object database)□□□□□□□□(tree object)□□□□□□□□□□(worktree)□□□□□□□□□□□(blob object)□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□(index)□HEAD□
```

cherry-picking

In SCM jargon, "cherry pick" means to choose a subset of changes out of a series of changes (typically commits) and record them as a new series of changes on top of a different codebase. In GIT, this is performed by the "git cherry-pick" command to extract the change introduced by an existing commit and to record it based on the tip of the current branch as a new commit.

cherry-picking

□SCM□□□□□“cherry pick“ □□□□□□□□□□□□□□□□□□(□□□□□)□□□□□□□□□□□()

clean[illegible]

[illegible](working tree)[illegible](current head)[illegible](clean)[illegible](dirty)[illegible]

commit

As a verb: The action of storing a new snapshot of the project's state in the git history, by creating a new commit representing the current state of the index and advancing HEAD to point at the new commit.

commit[illegible]

[illegible]git[illegible](commit)[illegible]"revision"[illegible]"version"[illegible](commit object)[illegible]

[illegible](commit)[illegible](index)[illegible] HEAD [illegible](snapshot)[illegible]git[illegible]

[illegible]

[illegible](particular revision)[illegible](tree object)[illegible]

core git

Git[illegible]

DAG

[illegible](commit objects)[illegible](direct parent)[illegible](chain)[illegible]

dangling object[illegible]

[illegible](unreachable object)[illegible](reference)[illegible](object)[illegible]

detached HEAD[illegible]HEAD[illegible]

[illegible]HEAD[illegible] git [illegible](commit)[illegible](the tip of any particular branch)[illegible]HEAD[illegible](detached)[illegible] [illegible].git/HEAD[illegible]SHA[illegible]

dircache

[illegible](index)[illegible]

directory[illegible]

[illegible]"ls"[illegible] :-)

dirty[illegible]

[illegible](dirty)[illegible]

ent

[illegible](tree-ish)[illegible]http://en.wikipedia.org/wiki/Ent_(Middle-earth)[illegible]

evil merge[illegible]

[illegible](parent)[illegible](evil merge)[illegible]

fast forward

A fast-forward is a special type of merge where you have a revision and you are "merging" another branch's changes that happen to be a descendant of what you have. In such these cases, you do not make a new merge commit but instead just update to his revision. This will happen frequently on a tracking branch of a remote repository.

[illegible]

“fast-forward”[illegible],()[illegible] [illegible](merge commit)[illegible] [illegible](remote repository)[illegible]

fetch[illegible]

[illegible](remote repository)[illegible]head ref[illegible] git fetch[illegible]

file system[illegible]

Linus Torvalds [illegible] git [illegible](user space)[illegible] infrastructure[illegible]git[illegible]

git archive

[illegible]

grafts

Grafts enables two otherwise different lines of development to be joined together by recording fake ancestry information for commits. This way you can make git pretend the set of parents a commit has is different from what was recorded when the commit was created. Configured via the `.git/info/grafts` file.

hash[illegible]

[illegible]git[illegible](object name)[illegible]

head

[illegible](named reference)[illegible](packed refs)[illegible]heads [illegible] `$GIT_DIR/refs/heads/`[illegible] [illegible]: git pack-refs

HEAD

[illegible](working tree)[illegible]HEAD[illegible]tree[illegible] HEAD [illegible]head[illegible]HEAD(detached HEAD)[illegible]

head ref

head[illegible]

hook

During the normal execution of several git commands, call-outs are made to optional scripts that allow a developer to add functionality or checking. Typically, the hooks allow for a command to be pre-verified and potentially aborted, and allow for a post-notification after the operation is done. The hook scripts are found in the `$GIT_DIR/hooks/` directory, and are enabled by simply removing the `.sample` suffix from the filename. In earlier versions of git you had to make them executable.

[illegible]

[illegible]git[illegible], () [illegible] ()

Typically[illegible]pre-verified[illegible]`$GIT_DIR/hooks/`[illegible]`.sample`[illegible]git[illegible]

index

A collection of files with stat information, whose contents are stored as objects. The index is a stored version of your working tree. Truth be told, it can also contain a second, and even a third version of a working tree, which are used when merging.

[illegible]

[illegible]()

index entry

The information regarding a particular file, stored in the index. An index entry can be unmerged, if a merge was started, but not yet finished (i.e. if the index contains multiple versions of that file).

[illegible]

[illegible] (master)

[illegible]git[illegible]"master"[illegible](active branch)[illegible]

merge

As a verb: To bring the contents of another branch (possibly from an external repository) into the current branch. In the case where the merged-in branch is from a different repository, this is done by first fetching the remote branch and then merging the result into the current branch. This combination of fetch and merge operations is called a pull. Merging is performed by an automatic process that identifies changes made since the branches diverged, and then applies all those changes together. In cases where changes conflict, manual intervention may be required to complete the merge.

merge[illegible]

[illegible]([illegible])[illegible]()

[illegible] fast forward[illegible](commit)[illegible](commit)[illegible](commit)[illegible]“[illegible]”(merge commit)[illegible]"[illegible]"(merge [illegible])[illegible]

object[illegible]

Git[illegible]SHA1[illegible]

object database[illegible]

[illegible](objects)[illegible](objects)[illegible] `$GIT_DIR/objects/`[illegible]

object identifier[illegible]

[illegible](object name)[illegible]

object name□□□□□

□□□□□□□□□□(unique identifier)□□□□□SHA1□□(Secure Hash Algorithm 1)□□□□□□□□(hash)□□□□□□□40□□□□□16□□□□□□□

object type□□□□□□

Git□4□□□□□□□□(commit)□□(tree)□□□(tag)□□□□□(blob)□

octopus□□□□

□□□□□□□□□□□(merge)□□□□□□□□□□□□□□

origin

□□□□□□□(upstream repository)□□□□□□□□□□□□□□(track)□□□(upstream)□□□□□□□ origin □□□□□□□□□□□□ ”□git branch -r`“ □□□□□□□□
(upstream repository)□□□□□□□□□ origin/name-of-upstream-branch □□□□□□(fetch)□□□□□□□□□□□

pack□□□

□□□□□□□□□□□□□□□□□(□□□□□□□□□□□□□□)□

pack index□□□□□

□(pack)□□□□□□□□□□□□□□□□□□□git□□□□□□(pack)□□□□□

parent

A commit object contains a (possibly empty) list of the logical predecessor(s) in the line of development, i.e. its parents.

□□□

□□□□□□(commit object)□()□

pickaxe

The term pickaxe refers to an option to the diffcore routines that help select changes that add or delete a given text string. With the `--pickaxe-all` option, it can be used to view the full changeset that introduced or removed, say, a particular line of text. See git diff.

plumbing

core git□□□(cute name)□

porcelain

Cute name for programs and program suites depending on core git, presenting a high level access to core git. Porcelains expose more of a SCM interface than the plumbing.

pull□□□

□(pull)□□□□□□□□□□□□(fetch)□□□□□(merge)□□□□□□□□□□□ git pull.

push

Pushing a branch means to get the branch's head ref from a remote repository, find out if it is a direct ancestor to the branch's local head ref, and in that case, putting all objects, which are reachable from the local head ref, and which are missing from the remote repository, into the remote object database, and updating the remote head ref. If the remote head is not an ancestor to the local head, the push fails.

□

()

reachable

All of the ancestors of a given commit are said to be "reachable" from that commit. More generally, one object is reachable from another if we can reach the one from the other by a chain that follows tags to whatever they tag, commits to their parents or trees, and trees to the trees or blobs that they contain.

□□□

rebase

□□□□(reapply)□□□(branch)□□□□□(base)□□□□□□□□□rebase□□□□□□□□□ head□ □□□□□□□□□□□□□□□□□□

ref□□□□

□□40□□□□SHA1□□□□□□□□□□□□□□□□□□□□□□ $GIT_DIR/refs/□

reflog

reflog□□□□□□□ref□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□3□□□(revision)□□□□□□□□□□□□□□□9□14□□□□□□□□□□□□□□□□□□□□□□□□□:git reflog□

refspec

"refspec"□□□□□□□□□□□□□□□□□ref□□□ref□□□□□□□□□□□□□□□□□:□□□□□□□□□□□□□□“+“□ □□□git fetch $URL refs/heads/master:refs/heads/origin □□□□□ $URL□□□□□□ head □□□□□□□□□origin□□□head□□git push $URL refs/heads/master:refs/heads/to-upstream □□□□□□□□□□□ head □□$URL□□ to-upstream□□□□□□□□□□□ git push□

repository

A collection of refs together with an object database containing all objects which are reachable from the refs, possibly accompanied by meta data from one or more porcelains. A repository can share an object database with other repositories via alternates mechanism.

resolve

□□□□□□□□□□□□□□□□□□□□□□

revision□□□□

□□□□(object database)□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□(commit object)□□□□

rewind

□□□□□□□□□□□□□□□head □□□□□□□□

SCM

[illegible]

SHA1

[illegible](object name)[illegible]

shallow repository

A shallow repository has an incomplete history some of whose commits have parents cauterized away (in other words, git is told to pretend that these commits do not have the parents, even though they are recorded in the commit object). This is sometimes useful when you are interested only in the recent history of a project even though the real history recorded in the upstream is much larger. A shallow repository is created by giving the `--depth` option to git clone, and its history can be later deepened with git fetch.

symref

Symbolic reference: instead of containing the SHA1 id itself, it is of the format 'ref: refs/some/thing' and when referenced, it recursively dereferences to this reference. 'HEAD' is a prime example of a symref. Symbolic references are manipulated with the git symbolic-ref command.

tag[illegible]

[illegible]ref[illegible] head [illegible]([illegible])[illegible]`$GIT_DIR/refs/tags/`[illegible] [illegible](commit ancerstry chain)[illegible]

tag object[illegible]

[illegible](ref)[illegible]PGP[illegible]“[illegible]”(signed tag object)[illegible]

topic branch

A regular git branch that is used by a developer to identify a conceptual line of development. Since branches are very easy and inexpensive, it is often desirable to have several small branches that each contain very well defined concepts or small incremental yet related changes.

tracking branch

A regular git branch that is used to follow changes from another repository. A tracking branch should not contain direct modifications or have local commits made to it. A tracking branch can usually be identified as the right-hand-side ref in a Pull:

`refspec.`

[illegible]

[illegible](follow)[illegible]git[illegible]()

tree[illegible]

[illegible](working tree)[illegible](tree object)[illegible]

tree object[illegible]

[illegible](list)[illegible](mode)[illegible](blob object)[illegible](tree object)[illegible](tree)[illegible]

tree-ish[illegible]

[illegible](commit object)[illegible](tree object)[illegible](tag object)[illegible](ref)[illegible]

unmerged index[illegible]

[illegible](index entries)[illegible]

unreachable object[illegible]

[illegible](reference)[illegible]

working tree[illegible]

[illegible](checkout)[illegible] [illegible] HEAD [illegible]