

# Git Internals

Jesús Espino García



**Kaleidos**  
beautiful code

11 de Abril de 2013

# ¿Por qué?

- La interfaz de git es de bajo nivel.
- Conocer git bien da mucho poder.
- El poder está ahí aunque no lo conozcamos.
- Un gran poder conlleva una gran responsabilidad.

# Conceptos básicos

- Procelain (Porcelana).
- Plumbing (Cañerías).
- Objetos
- Referencias
- Head
- Working copy

# Contenido de .git

- Ficheros.
  - HEAD
  - index
  - config
- Directorios.
  - objects
  - refs
  - hooks
  - info

# Objetos

- Bloque de datos almacenado en git
- Referenciado por el sha1 de su contenido
- Almacenados en el directorio `.git/objects/` (o en packs).
- Hay 4 tipos de objetos en git (blob, tree, commit, tag).

# blobs

- Será el nodo hoja de nuestros arboles.
- Será equivalente (normalmente) a nuestros ficheros.

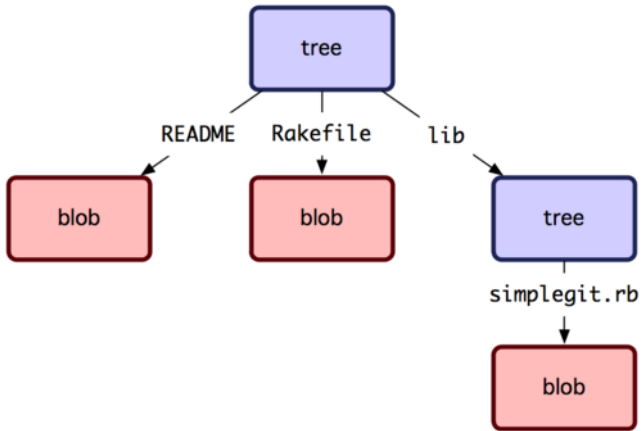
## Ejemplos de almacenar blob

- `git init repo; cd repo`
- `echo 'version 1' | git hash-object -w --stdin`
- `find .git/objects -type f`
- `git cat-file -p 83baae61804e65cc73a7201a7252750c76066a30`
- `echo 'version 2' | git hash-object -w --stdin`
- `find .git/objects -type f`
- `git cat-file -p 1f7a7a472abf3dd9643fd615f6da379c4acb3e3a`

# trees

- Es un directorio de referencias a blob y otros trees.
- Almacena referencias (sha1 de objetos) y metadatos.

## diagrama de ejemplo de trees



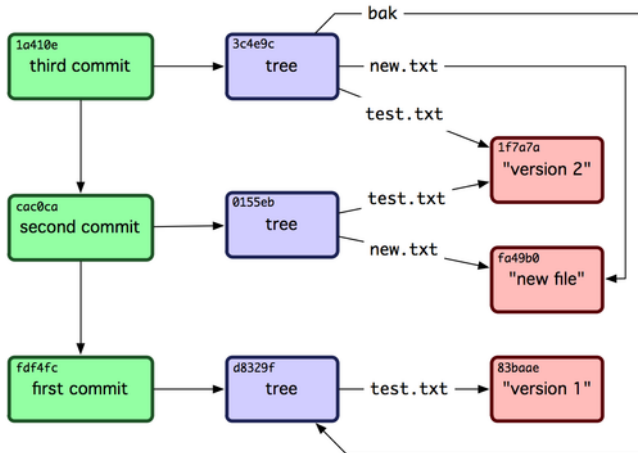
## Ejemplos de almacenar tree

- `git init repo; cd repo`
- `git update-index --add --cacheinfo 100644 \  
83baae61804e65cc73a7201a7252750c76066a30 test.txt`
- `git write-tree`
- `find .git/objects -type f`
- `git cat-file -p d8329fc1cc938780ffdd9f94e0d364e0ea74f579`

# commits

- Almacena una referencia a un tree.
- Almacena una referencia a su commit padre.
- Almacena metadatos del commit (autor, fecha, mensaje...)

## diagrama de ejemplo de commits



## Ejemplos de commit

- `git init repo; cd repo`
- `echo "Version 1" > fichero.txt; git add fichero.txt`
- `git commit -m "Version 1"`
- `find .git/objects -type f`
- `git cat-file -p HEAD`
- `git cat-file -p b7c0dba424af1e98a3570f8125476126129e5c32`
- `git cat-file -p fb8247c7b27ae4cad9e7e3e66ba95126658ea7c2`

# tags

- Almacena una referencia a un commit.
- Almacena metadatos del commit (autor, fecha, nombre...)

# Objects storage

- Se añade una cabecera con el tipo de objeto y la longitud del mismo.
- Se concatena con los datos que se van a almacenar
- Se calcula su sha1 que se utilizara como nombre del objeto.
- Se comprime con zlib.
- Se almacena en `.git/objects/XX/XXXXXX...`

## Ejemplos de lectura directa

- `git init repo; cd repo`
- `echo "Version 1" > fichero.txt; git add fichero.txt`
- `git commit -m "Version 1"`
- `find .git/objects -type f`
- `git cat-file -p HEAD`
- `git cat-file -p b7c0dba424af1e98a3570f8125476126129e5c32`
- `git cat-file -p fb8247c7b27ae4cad9e7e3e66ba95126658ea7c2`
- `cat .git/objects/fb/8247c7b27ae4cad9e7e3e66ba95126658ea7c2 \`  
`| zlib-flate -uncompress`

# Packfiles

- Paquetes de objetos.
- Periodicamente git empaqueta los objetos en packs (gc).
- Se almacenan en `.git/objects/pack/`.
- Hay un listado de packs en `.git/objects/info/packs`.
- Cada pack tiene su índice en `.git/objects/pack/`.

# Referencias

- Están en `.git/refs` principalmente.
- Son punteros a objetos.
- Contienen el id del objeto al que apuntan.

# branches

- Están en `.git/refs/heads`

# HEAD

- Esta en `.git/HEAD`
- Es una referencia simbolica que apunta a la referencia de la rama actual.

# tags

- Están en `.git/refs/tags`
- Son referencias a commits o a objetos tag.
- Los tags simples son una referencia directa a un commit.
- Los tags con anotaciones son referencias a un objeto tag que apunta a un commit.

# remotes

- Están en `.git/refs/remotes`
- Contiene las referencias de mis remotes.
- Se actualizan cuando hago push o fetch.

# Refspects

- Forma de definir la relación entre las referencias de diferentes orígenes
- Tienen el formato `[+]<src>:<dst>`
- Opcionalmente se pone un `+` para actualizar la referencia cuando no hay fast-forward.
- Las referencias pueden tener `*` para definir "todo lo que haya en un directorio"
- No se permite el uso de `*` para selecciones parciales de referencias.
- Ejemplo: `+refs/heads/*:refs/remotes/origin/*`

# Pulling with refspects

- Ejemplo: `git fetch origin master:refs/remotes/origin/mymaster`
- Descarga la referencia master del origin a refs/remotes/origin/mymaster en local

# Pushing with refspects

- Ejemplo: `git push origin master:refs/heads/qa/master`
- Envía la referencia master local al refs/heads/qa/master en origin

# Borrando referencias

- Ejemplo: `git push origin :topic`
- Borra la referencia topic en origin

# init

Crea un .git con datos básicos.

- Un fichero config por defecto
- Los directorios refs, objects e info
- Un HEAD apuntando a la referencia master
- Y poco más

# add

- añade el fichero a la base de datos de Objetos
- añade el fichero al index

# commit

- Añade el tree apuntando al arbol de ficheros al que apunte el index a la base de datos de Objetos
- Añade el commit apunte al tree recién añadido a la base de datos de objetos
- modifica el HEAD
- modifica la referencia de la rama actual

# ¿De qué no hemos hablado?

- Más comandos de porcelana.
- Comandos de plumbing.
- Protocolos de transferencia.
- Mantenimiento y recuperación de datos.
- Stash

# Referencias

- <http://git-scm.com/> - Web oficial de git.
- <http://git-scm.com/book> - ProGit (El libro de Git).
- <http://gitguys.com/> - Página sobre git.
- <http://github.com/> - Servicio de git por excelencia.
- <http://bitbucket.org/> - Servicio de git de repositorios privados gratis.

# Dudas

...