

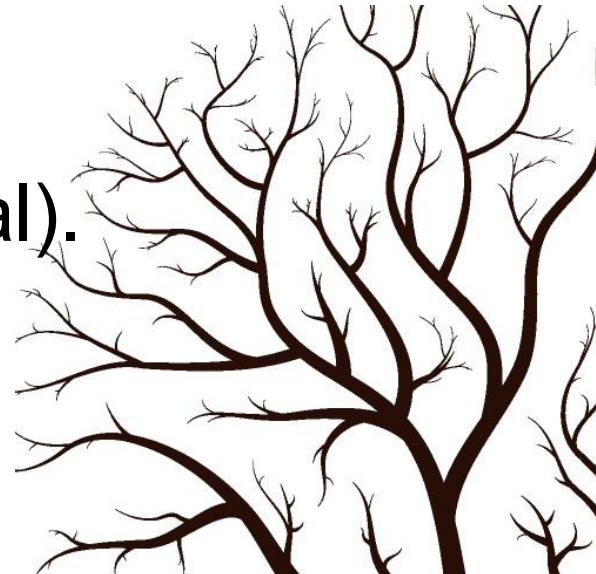


git

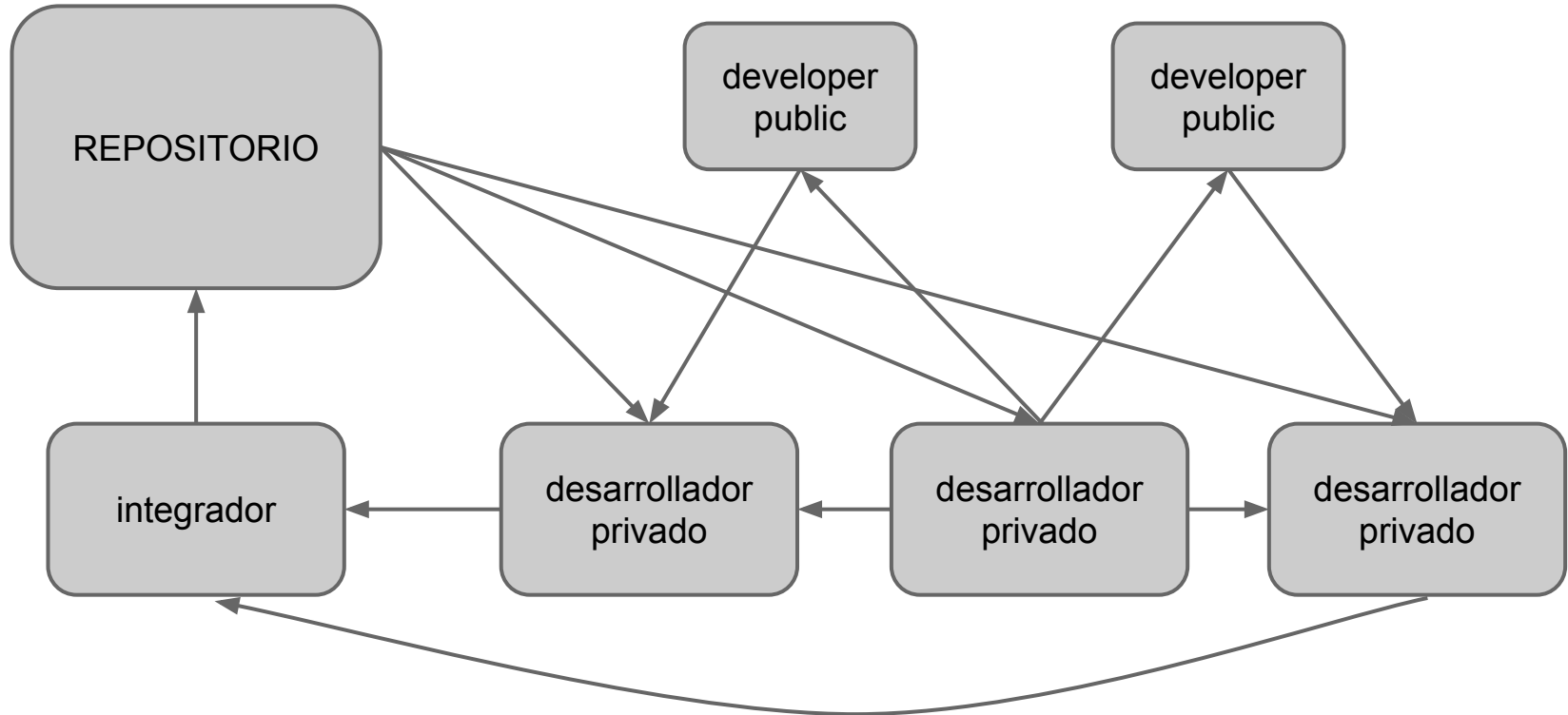
Curso introductorio

¿Qué es Git?

- Programa CLI que funciona con archivos.
- Control de versiones distribuido
- Orientado a contenido
- Es muy rápido!
- Control de Acceso (commit local).
- Ramas, ramas, RAMAS!!



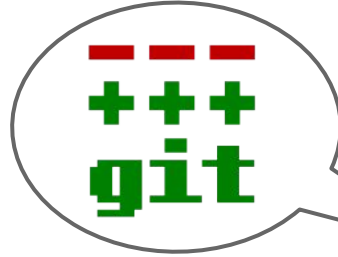
¿Qué es Git?



Orígenes...



Linux™



Github no es git...

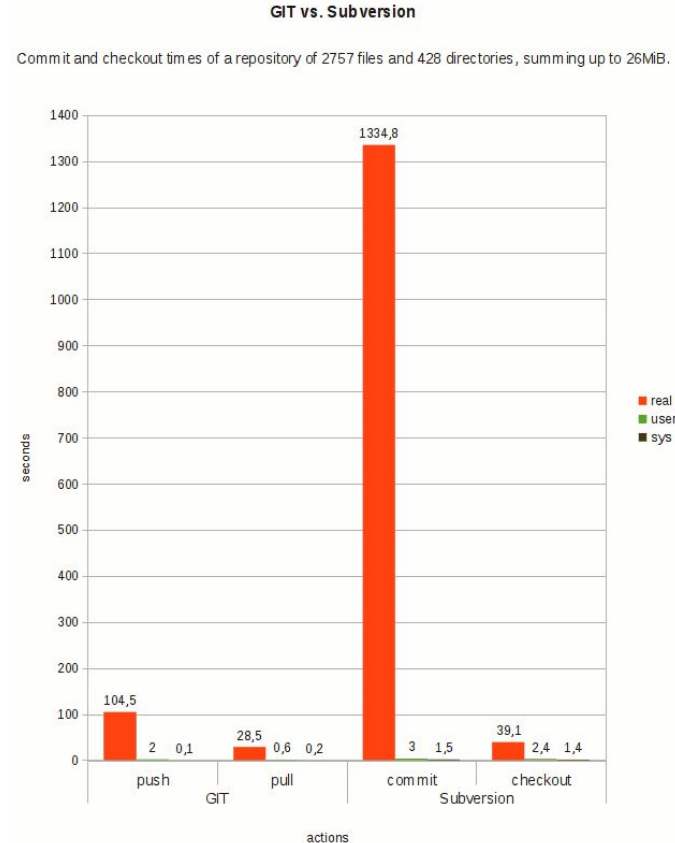


git

contenido... RAPIDEZ + COMPACTO.

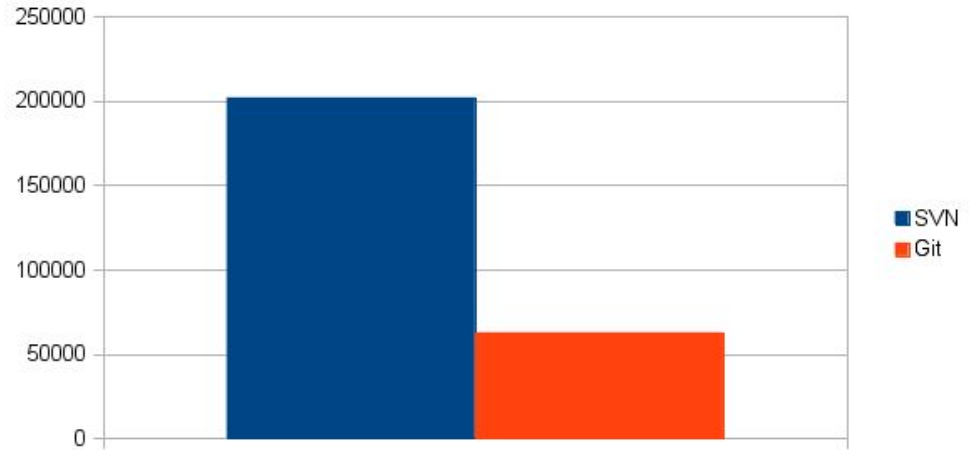
- solo se transfieren **cambios/deltas** en lugar de ficheros completos, lo que implica operaciones de red mucho más rápidas.
- solo se almacenan **cambios/deltas** en lugar de ficheros completos, lo que implica repositorios mucho más compactos.

<http://www.lourdas.name/blog/git-vs-subversion-performance-comparison>
<http://vcs.atSPACE.co.uk/2012/11/05/which-repository-is-more-compact-git-or-svn/>



contenido... RAPIDEZ + COMPACTO.

- solo se transfieren **cambios/deltas** en lugar de ficheros completos, lo que implica operaciones de red mucho más rápidas.
- solo se almacenan **cambios/deltas** en lugar de ficheros completos, lo que implica repositorios mucho más compactos.



SVN repository size vs Git repository size

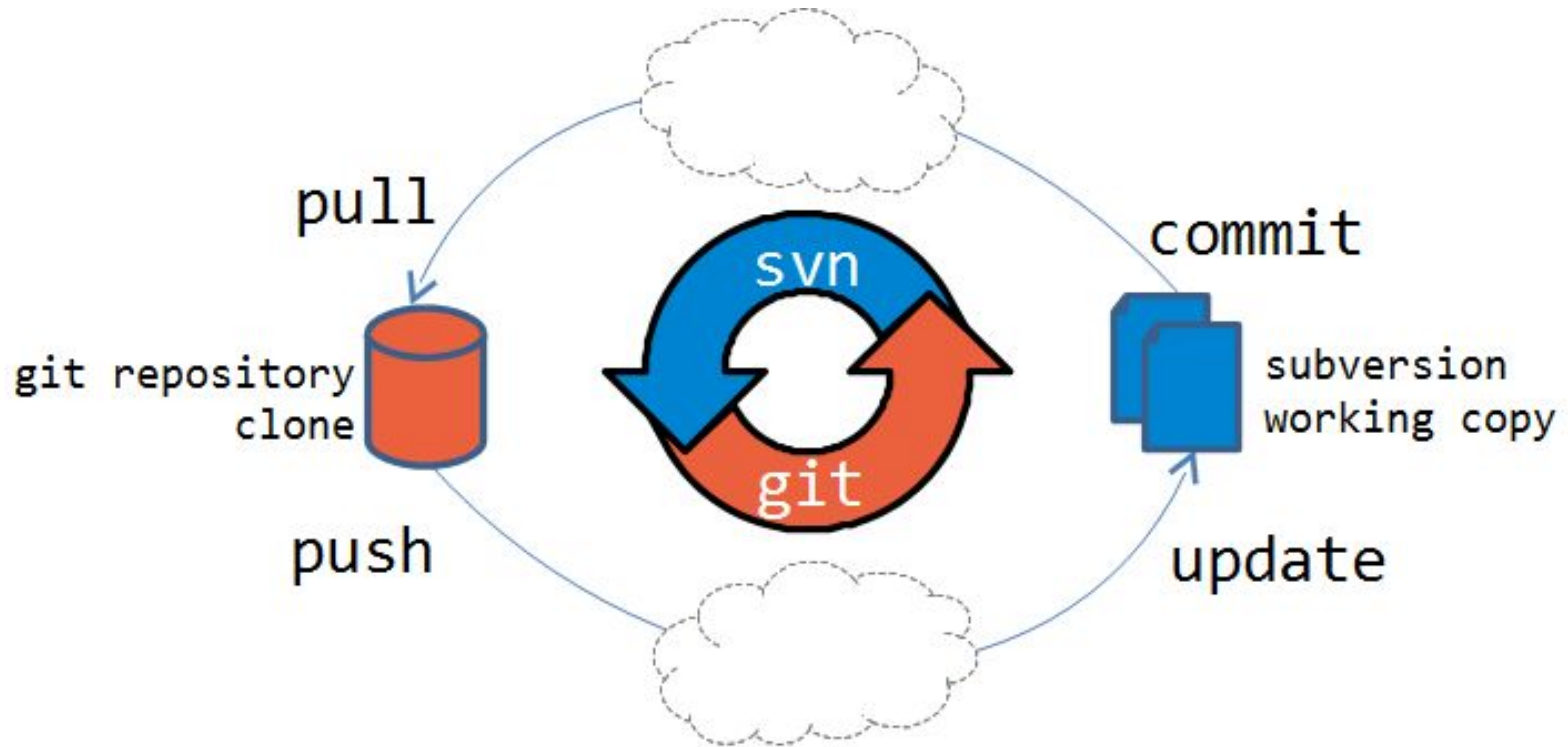
Argggghh merging!



Como git trabaja con contenido y no con archivos es más eficiente por que:

- busca rápidamente dentro de su grafo lo que necesita ser integrado (merge).
- aplica solo el parche correcto (únicamente deltas).

Comparación Git - SVN



me gusta más svn....

1. GIT tiene modelo de información **complejo**...
2. en GIT las tareas simples necesitas muchos **más pasos/comandos** que con SVN.
3. SVN repositorio **único** y **central**...



Instalación y configuración



...OSX ya viene por defecto

... Ubuntu, apt-get install git-core git-doc

...Windows: <https://git-scm.com/download/win>

...otros, <http://git-scm.com/download>

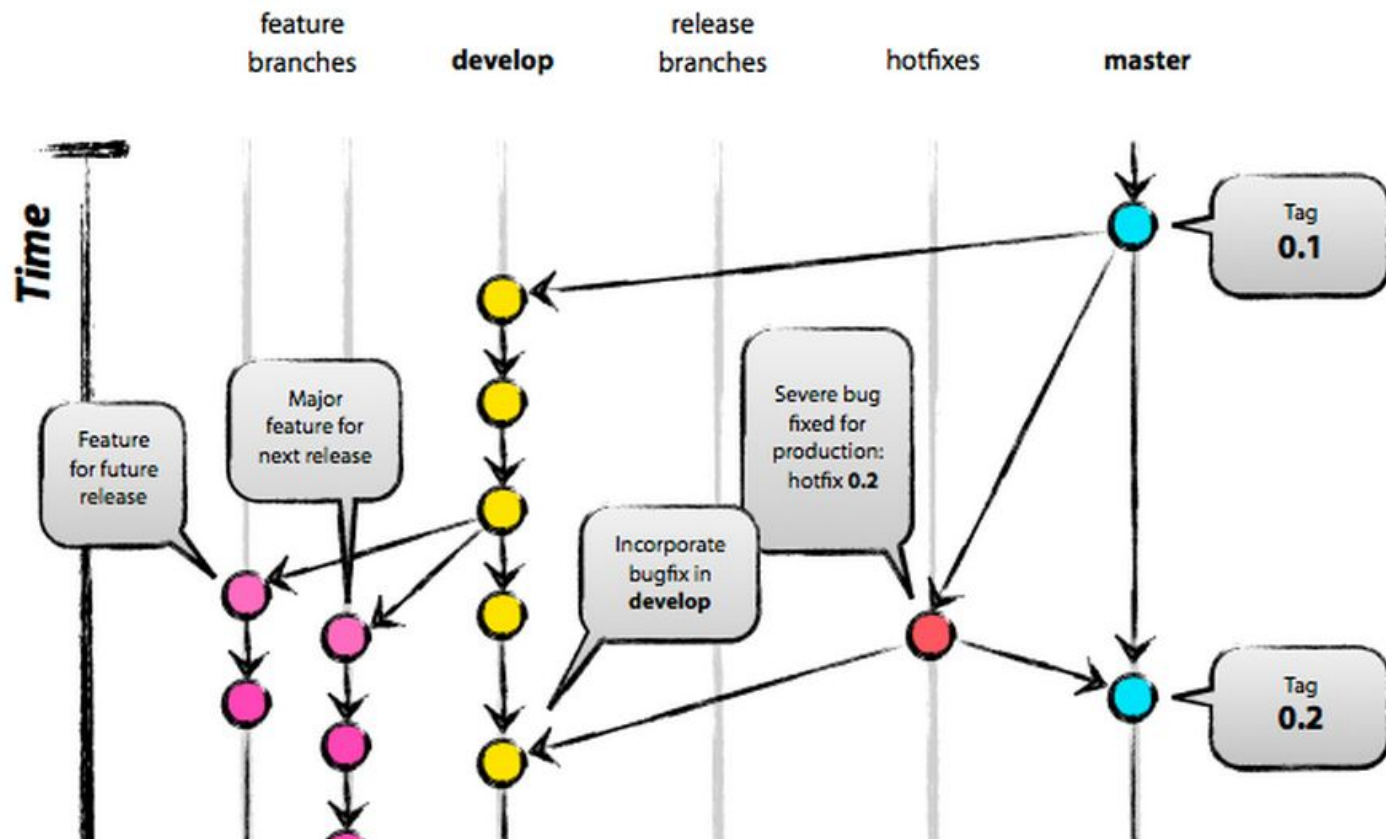
```
$ git config --global user.name "Luis Gonzalez"
```

```
$ git config --global user.email luis.gonzalez@beeva.com
```

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

```
$ git config --list
```

Git FLOW!



basics: init / clone / .gitignore

\$ git init

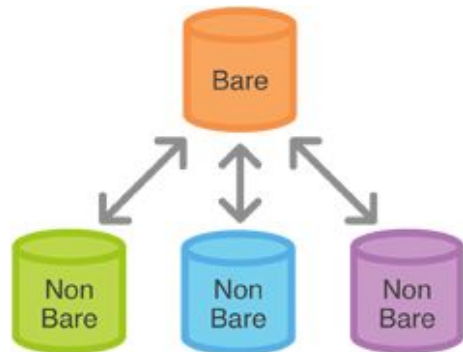
Inicializa el directorio actual

\$ git init <directory>

Inicializa el directorio objetivo

\$ git init --bare <directory>

lo inicializa como “bare”



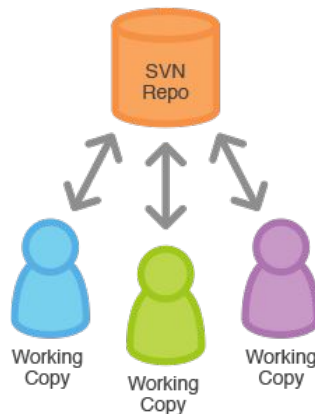
\$ git clone <repo>

clona el repo con mismo nombre

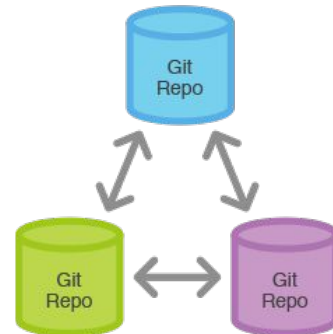
\$ git clone <repo> <directory>

clona repo con otro nombre local

Central-Repo-to-Working-Copy
Collaboration



Repo-to-Repo
Collaboration



basics: status / add / commit

STATUS

\$ git status

On branch **master**

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes)

#

modified: example1.js

#

Untracked files:

(use "git add <file>..." to include in what will be committed)

#

nuevo.js

ADD

\$ git add <file>

\$ git add <directory>

COMMIT

\$ git commit

\$ git commit -m "message"

\$ git commit -a

basics: rm / mv / log / diff

RM - quitar del control de versiones de git...

```
$ git rm <file>
```

MV - renombrar ficheros o carpetas...

```
$ git mv <source> <destination>
```

```
$ git mv <source> ... <destination directory>
```

LOG

```
$ git log -3 -p
```

DIFF

```
$ git diff
```

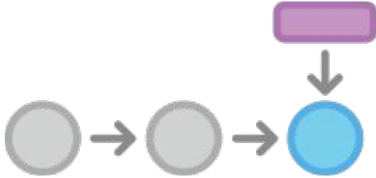
```
$ git diff <commit> <commit>
```

branching: branch / merge / checkout

BRANCH

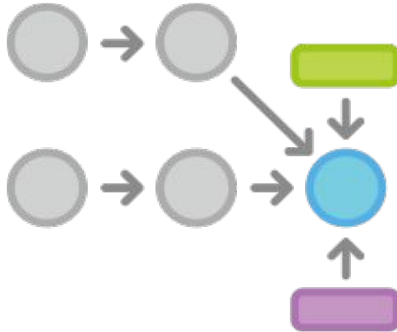
\$ git branch

\$ git branch <branchname>



MERGE

\$ git merge <branch>

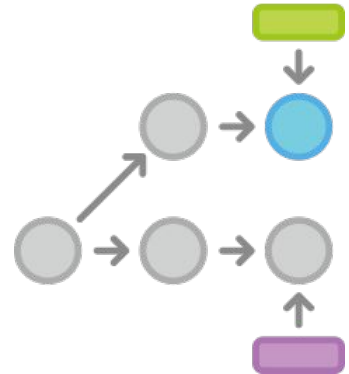


CHECKOUT

\$ git checkout <existing-branch>

\$ git checkout -b <new-branch>

\$ git checkout -b <new> <existing>



branching: tag

\$ git tag

lista

\$ git tag v1.4

referencia a un branch

\$ git tag -a v1.4 -m 'version 1.4'

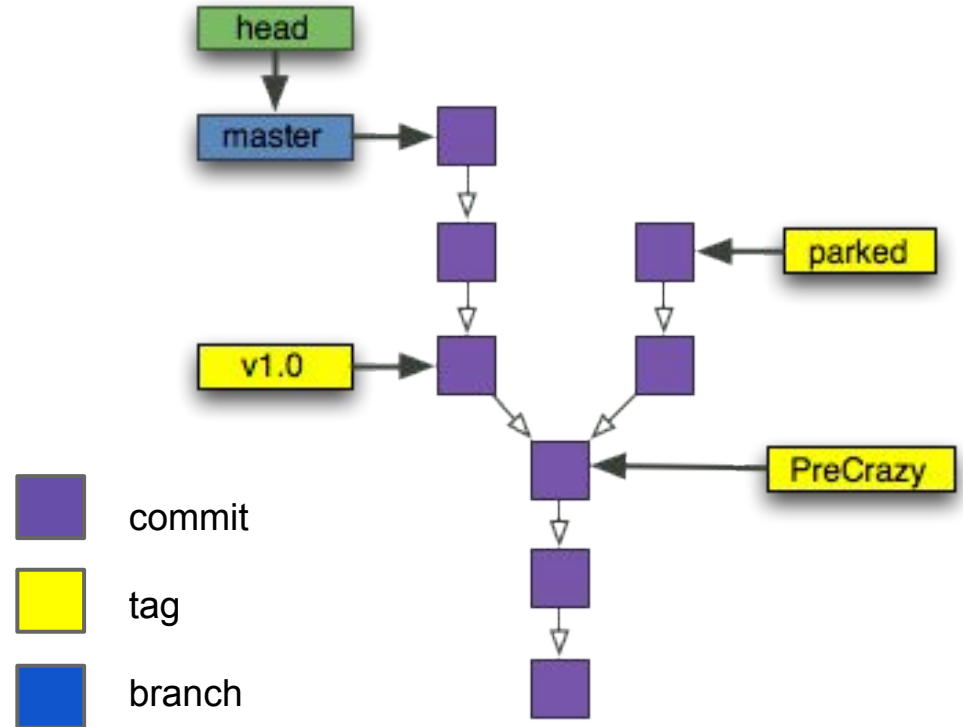
checksum, objetos completos

\$ git tag -s v1.4 -m 'version 1.4'

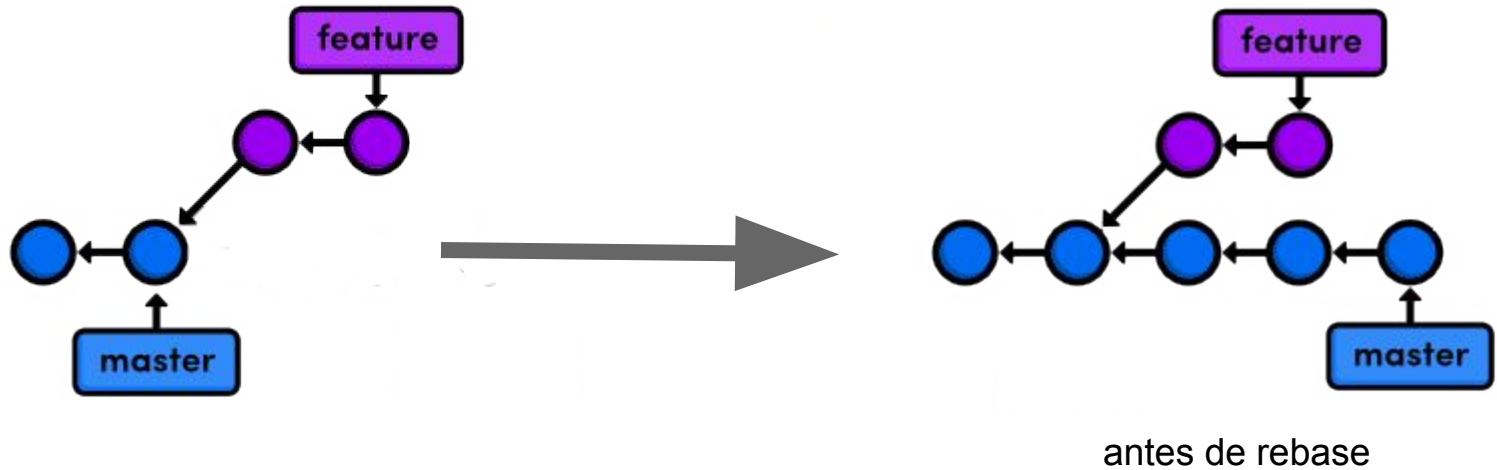
firmado con GPG

\$ git show v1.4

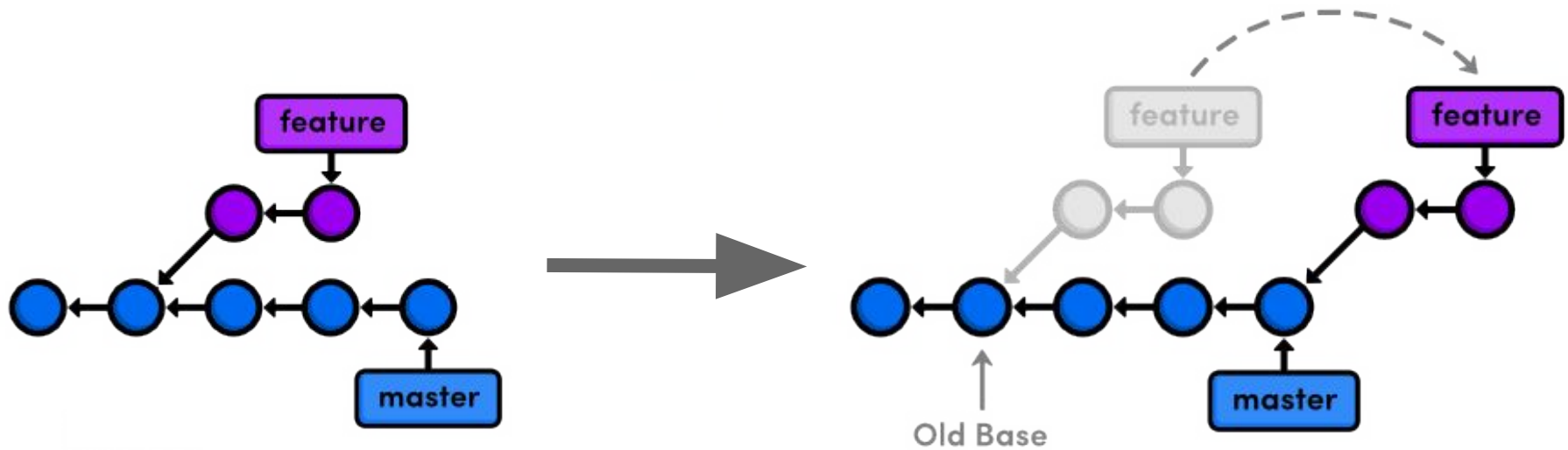
ver un tag



branching: rebase



branching: rebase

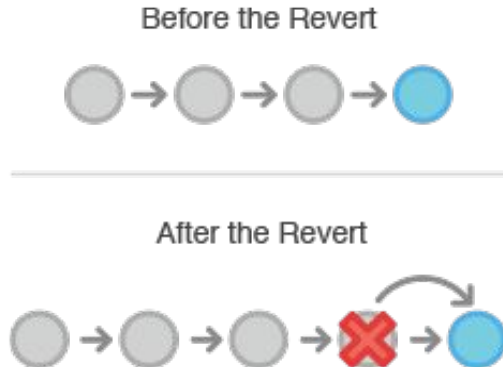


después de rebase

undoing changes...

REVERT

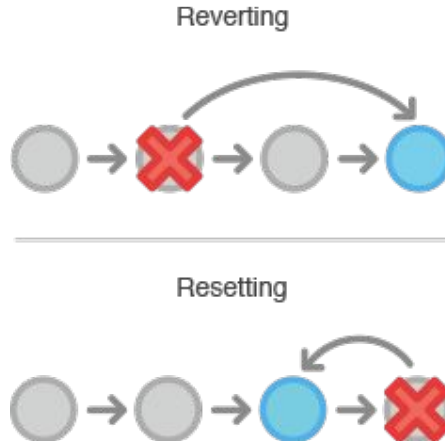
`$ git revert <commit>`



RESET

`$ git reset`

`$ git reset <file>`



CLEAN

`$ git clean`



distributed: origin / master / remotes

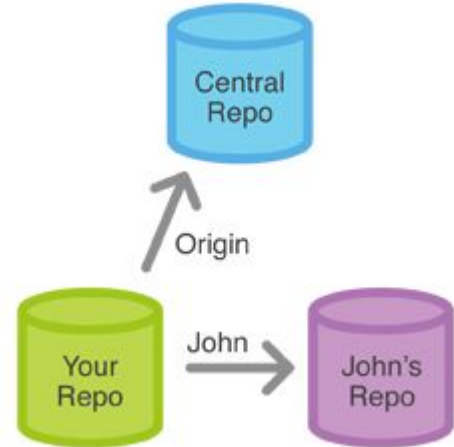
Master es la rama principal de todo repositorio git (equivalente al **trunk**).

Origin es el remote principal de un repositorio **clonado**.

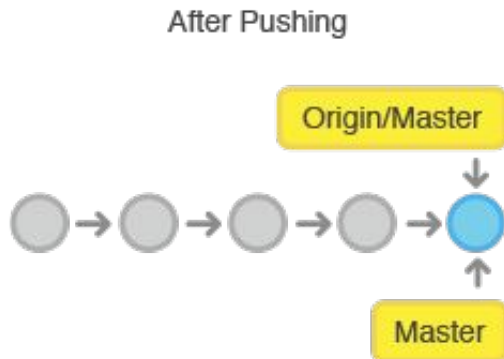
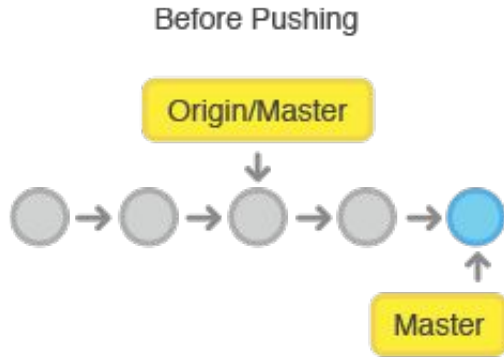
\$git remote add <remotename> <uri>

\$git remote rm <remotename>

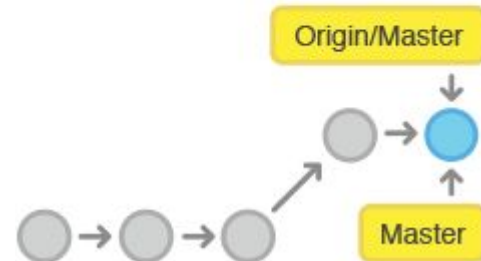
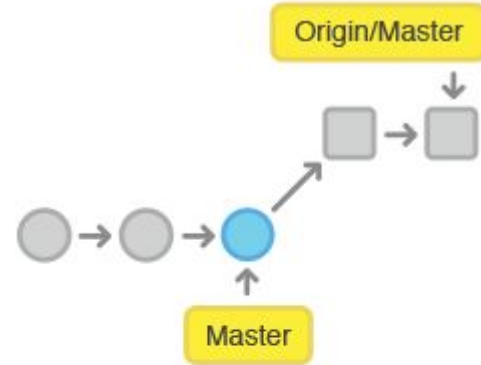
\$git remote mv <remotename> <newname>



distributed



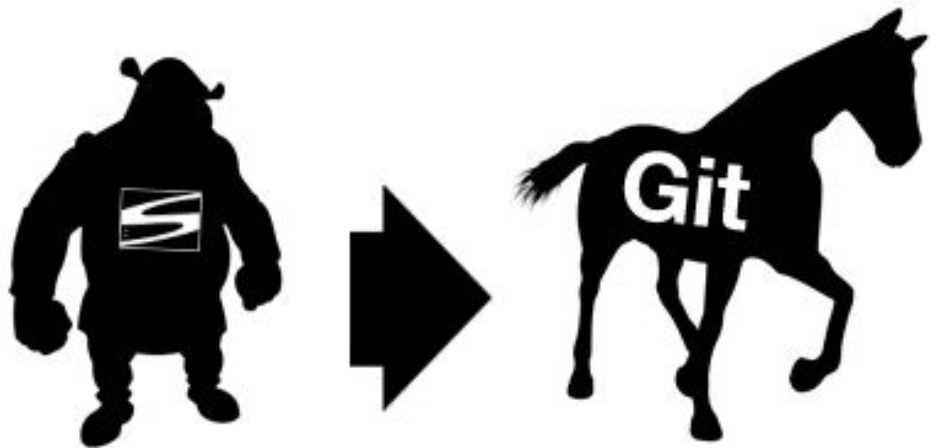
`$ git push/pull <remote> <branch>`



advanced: git-svn

luis@boxita:~\$ git svn

blame	clone	create-ignore
fetch	gc	init
migrate	propget	rebase
set-tree	show-ignore	branch
commit-diff	dcommit	find-rev
info	log	mkdirs
proplist	reset	tag



advanced: hooks

```
#!/bin/sh
# Mensajito
echo "Publicando rama master!" >&2
# borrar el website anterior
rm -rf /var/www/my-website
# Crea el directorio
mkdir /var/www/my-website
# genera el tar de MASTER
git archive master --format=tar --output=/var/www/my-website.tar
# Lo descomprime en el directorio deploy
tar -xf /var/www/my-website.tar -C /var/www/my-website
exit 0
```



advanced: stash

commits en-sucio...

\$ git stash list -----> lista de parches en la pila

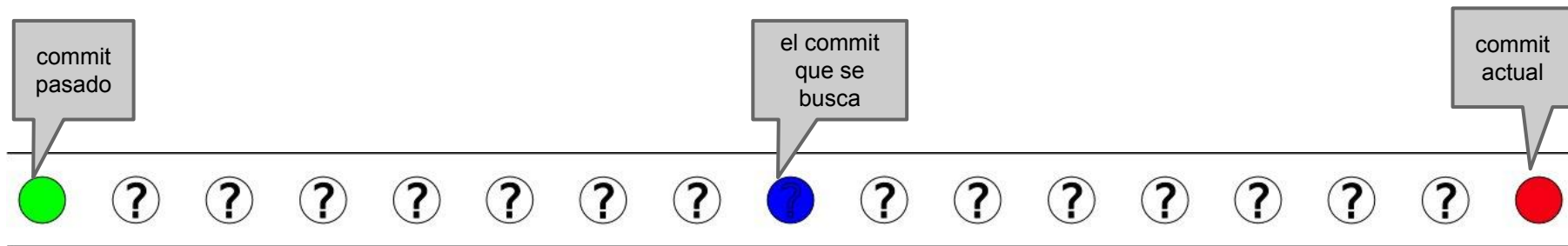
\$ git stash -----> crea un parche

\$ git stash pop -----> aplica el último parche

\$ git stash apply -----> aplica todos los parches de la pila.

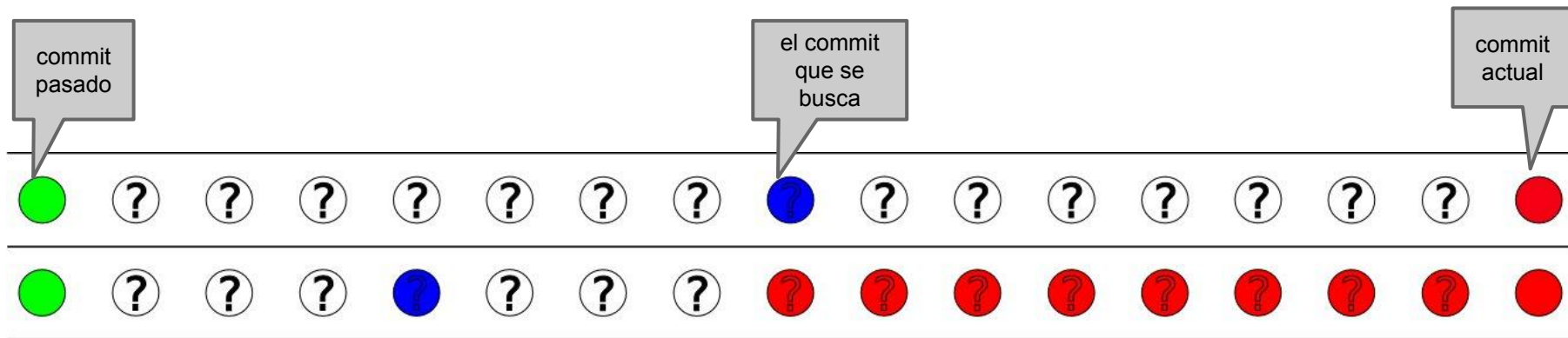
advanced: bisection

Búsqueda binaria de cuando (y quien) se inyectó un defecto en un commit.



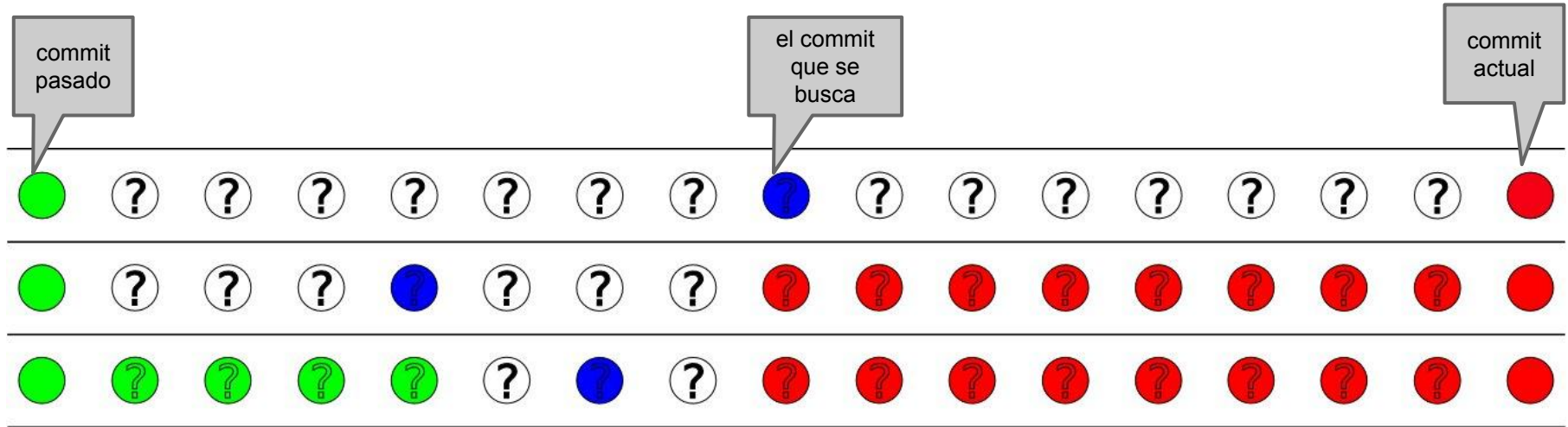
advanced: bisection

Búsqueda binaria de cuando (y quien) se inyectó un defecto en un commit.



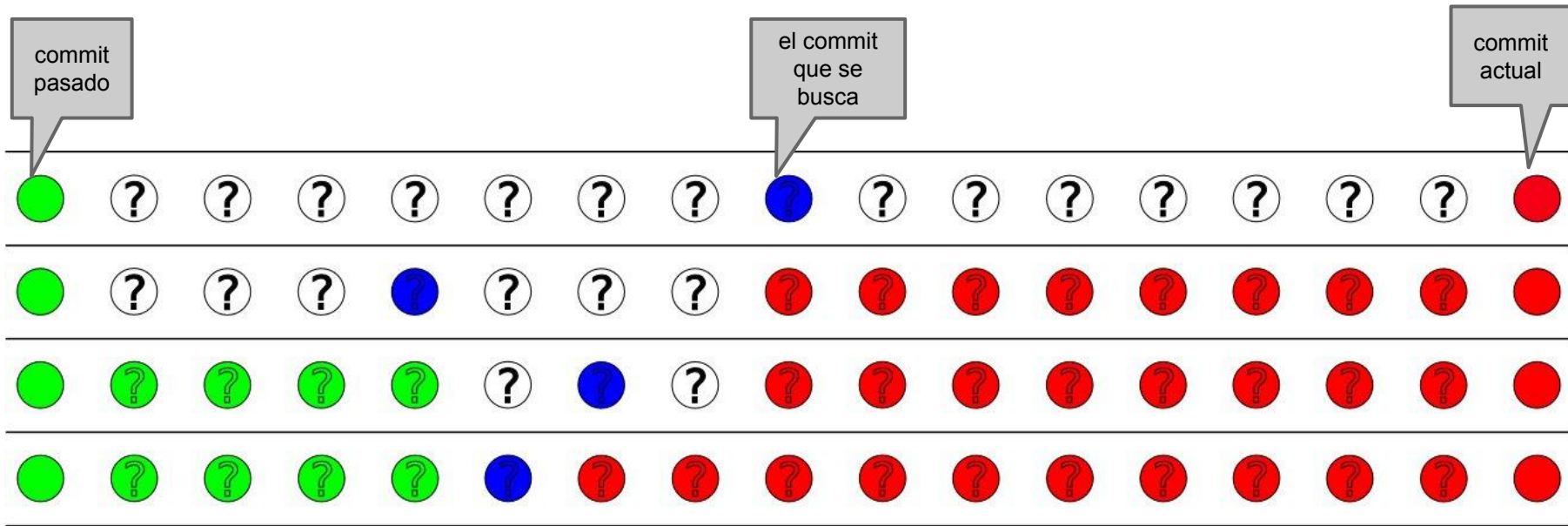
advanced: bisection

Búsqueda binaria de cuando (y quien) se inyectó un defecto en un commit.



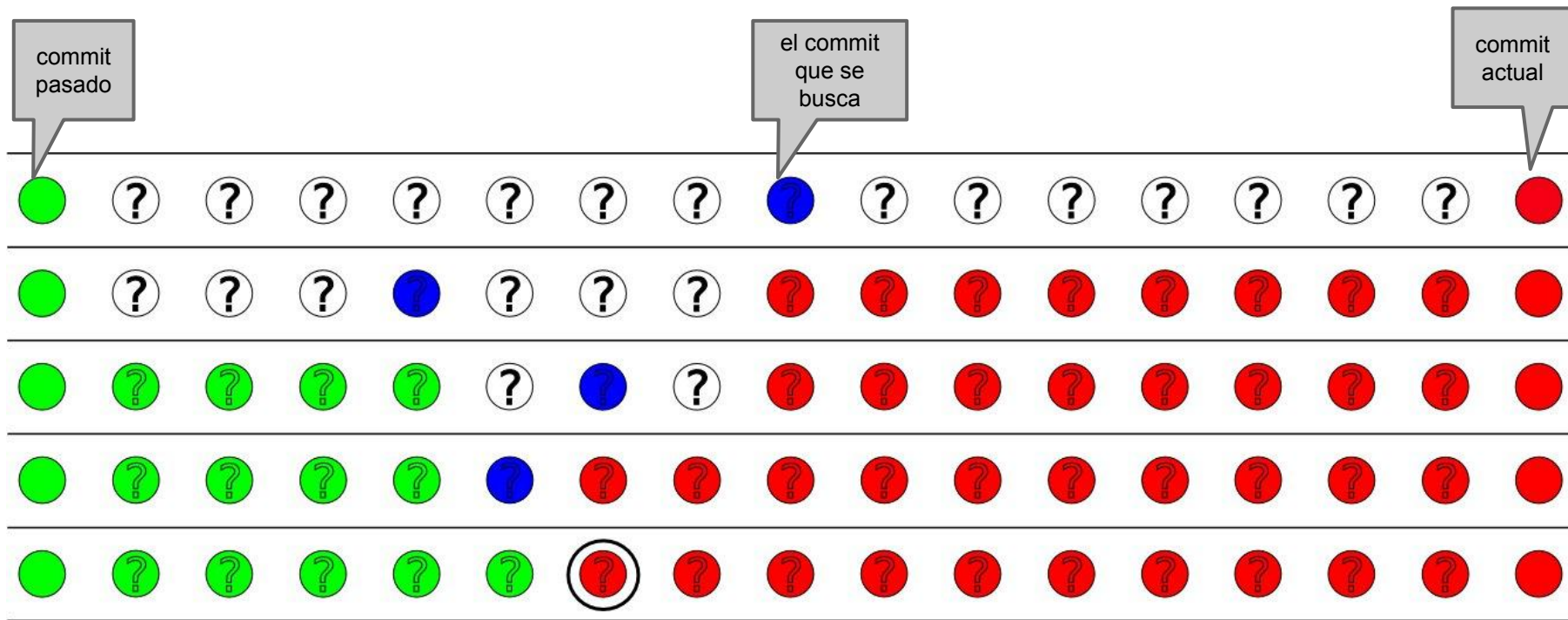
advanced: bisect

Búsqueda binaria de cuando (y quien) se inyectó un defecto en un commit.



advanced: bisection

Búsqueda binaria de cuando (y quien) se inyectó un defecto en un commit.

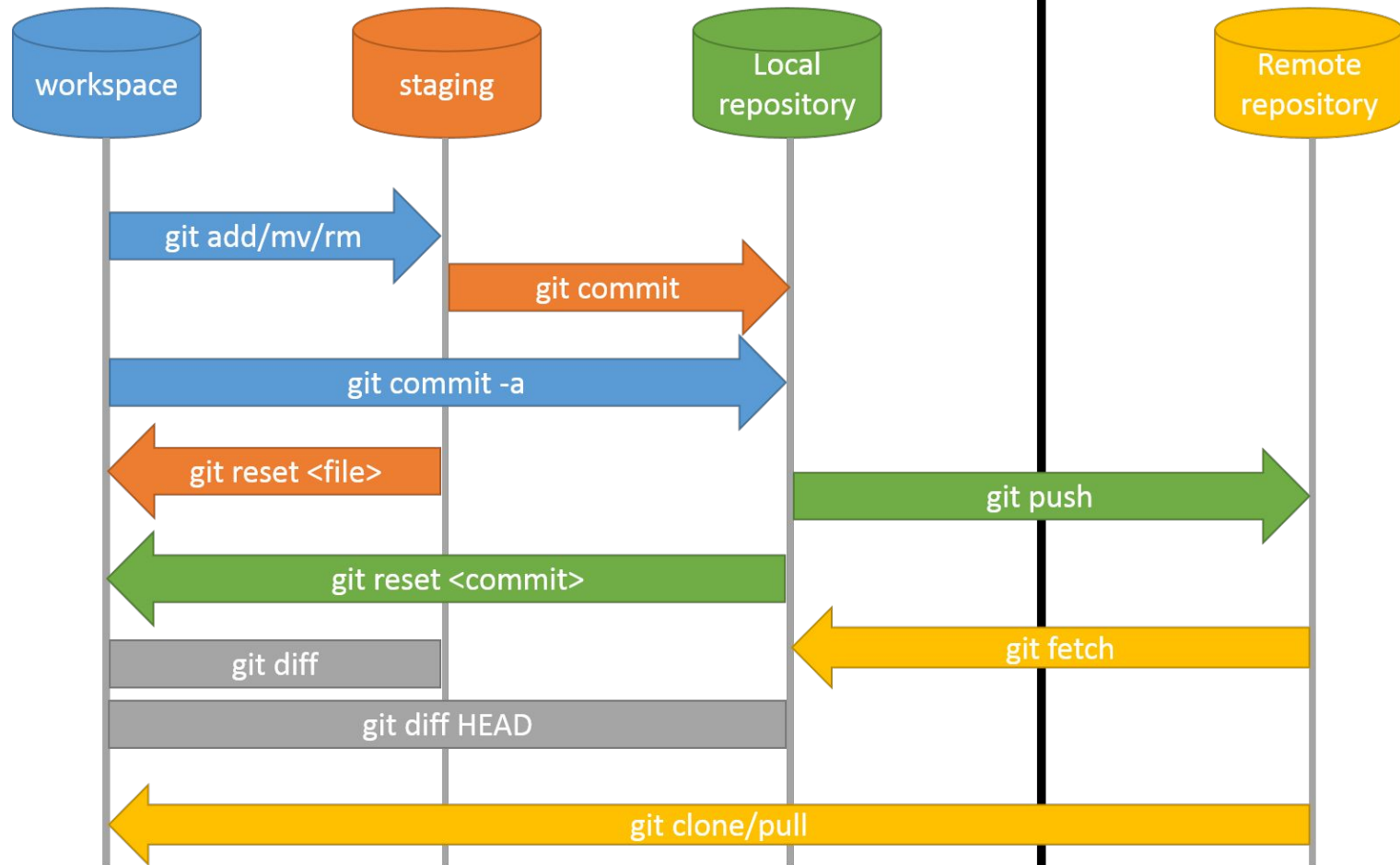


advanced: blame

Time	Author	Commit Hash	File Path	Line Number	Code Snippet
2009-01-16 14:27	Jonas Fonseca	4bb9536	src/tig.c	165	};
2009-01-16 14:27	Jonas Fonseca	4bb9536	src/tig.c	166	typedef enum input_status (*i-
2009-01-16 14:27	Jonas Fonseca	4bb9536	src/tig.c	167	static char *prompt_input(con-
2009-01-16 14:27	Jonas Fonseca	4bb9536	src/tig.c	168	static bool prompt_yesno(con-
2006-05-15 03:50	Jonas Fonseca	6706b2b	src/tig.c	169	struct menu_item {
2009-02-22 01:19	Jonas Fonseca	91e8041	src/tig.c	170	int hotkey;
2009-02-22 01:19	Jonas Fonseca	91e8041	src/tig.c	171	const char *text;
2009-02-22 01:19	Jonas Fonseca	91e8041	src/tig.c	172	void *data;
2009-02-22 01:19	Jonas Fonseca	91e8041	src/tig.c	173	};
2009-02-22 01:19	Jonas Fonseca	91e8041	src/tig.c	174	static bool prompt_menu(const-
2009-02-22 01:19	Jonas Fonseca	91e8041	src/tig.c	175	/*
2009-02-22 01:19	Jonas Fonseca	91e8041	src/tig.c	176	* Allocation helpers ... Ent-
2009-02-22 01:19	Jonas Fonseca	91e8041	src/tig.c	177	*/
2006-05-10 22:16	Jonas Fonseca	03a93db	src/tig.c	178	#define DEFINE_ALLOCATOR(name
2009-02-18 11:47	Jonas Fonseca	b2ff9a4	src/tig.c	179	static type *
2009-02-18 11:47	Jonas Fonseca	b2ff9a4	src/tig.c	180	
2009-02-18 11:47	Jonas Fonseca	b2ff9a4	src/tig.c	181	
2009-02-18 11:47	Jonas Fonseca	b2ff9a4	src/tig.c	182	
2009-02-18 11:47	Jonas Fonseca	b2ff9a4	src/tig.c	183	

[blame] tig.c - line 172 of 7494 (2%)

Source file



cheatsheets

<https://git.wiki.kernel.org/images-git/7/78/Git-svn-cheatsheet.pdf>

https://na1.salesforce.com/help/pdfs/en/salesforce_git_developer_cheatsheet.pdf

<http://ndpsoftware.com/git-cheatsheet.html>

http://www.git-tower.com/blog/assets/git-cheatsheet/Git_Cheat_Sheet_all.zip

<http://rypress.com/tutorials/git/remotes.html>

<https://github.com/hadesbox/curso-basico-git>

1. instalar git (<http://git-scm.com/downloads>)
2. \$ git config --global user.name "Luis Gonzalez"
\$ git config --global user.email luis.gonzalez@keeper.io
\$ git config --list
3. hacer los ejercicios 1, 2 y 3 en local.