BEEVA

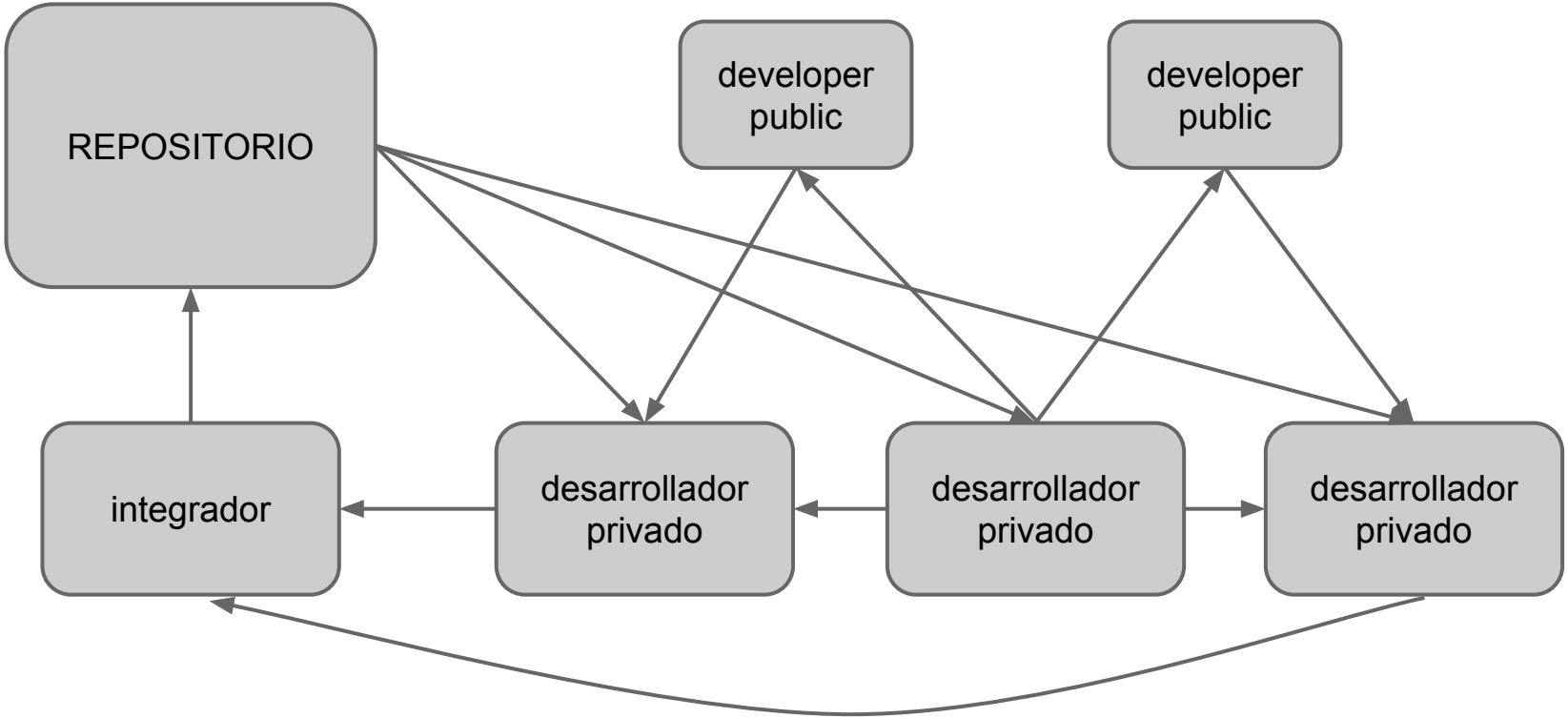
git

Curso introductorio,
Beeva 2016.

¿Qué es Git?

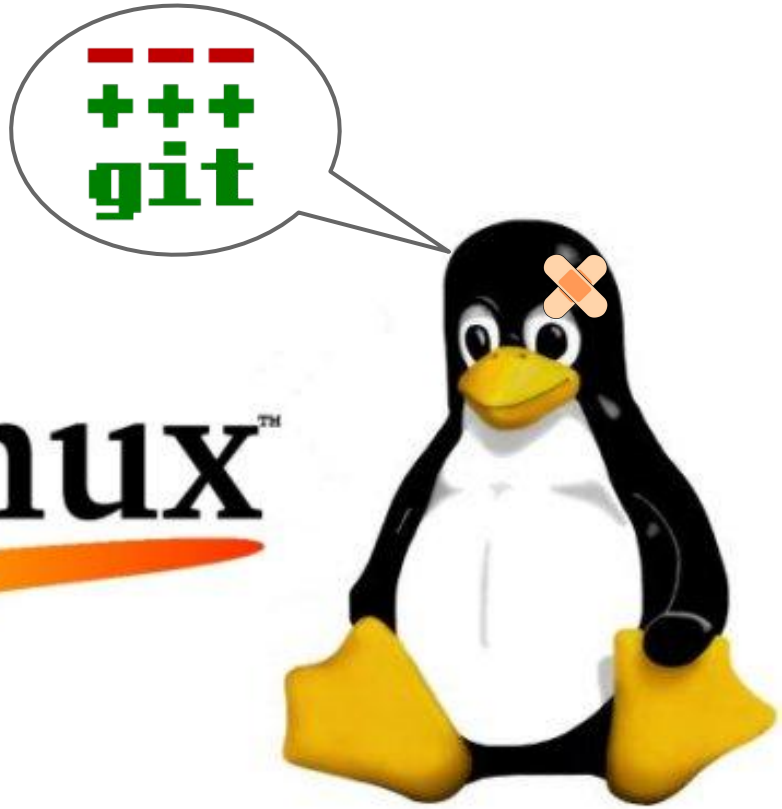
- Programa CLI que funciona con archivos.
- Control de versiones distribuido
- Orientado a contenido
- Es muy rápido!
- Control de Acceso (commit local).
- Ramas, ramas, RAMAS!!

¿Qué es Git?


REPOSITORIO
developer public
developer public
integrador
desarrollador privado
desarrollador privado
desarrollador privado

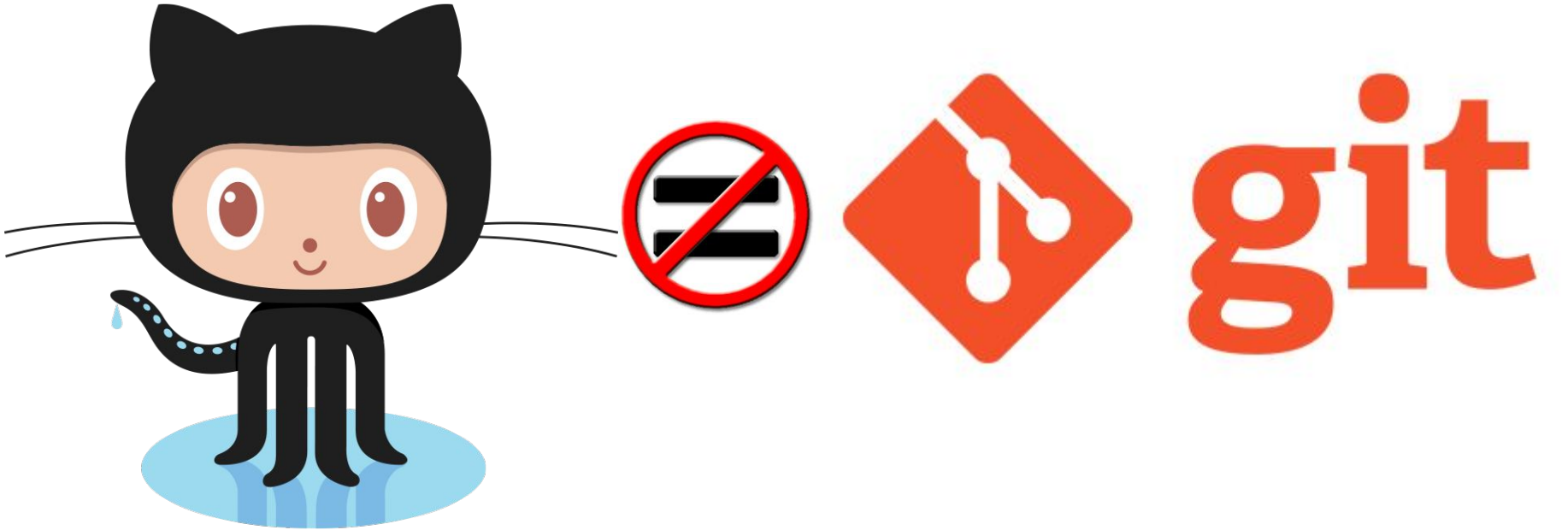
BEEVA

Orígenes...



BEEVA

Github no es git...

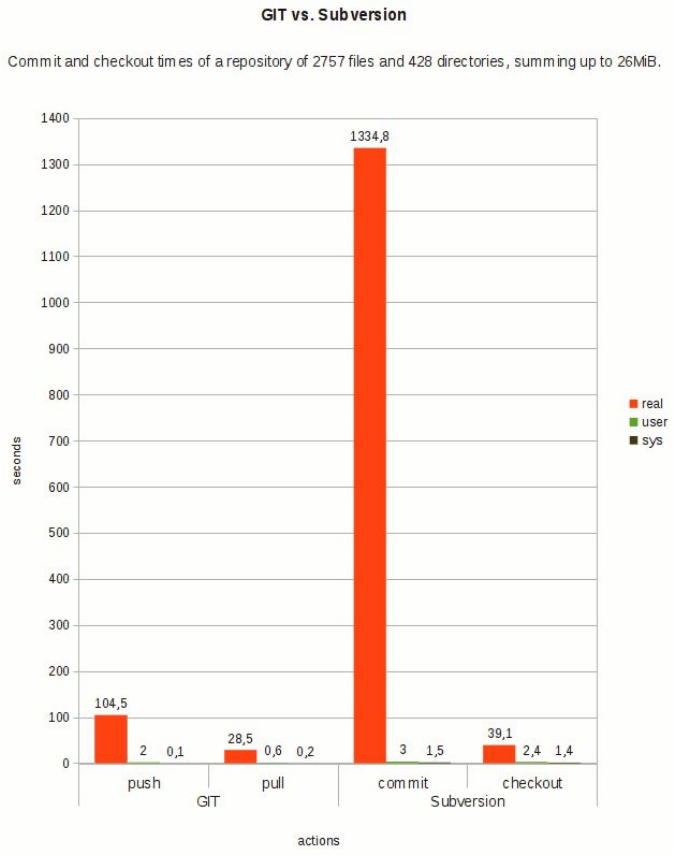


contenido… RAPIDEZ + COMPACTO.

- 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.

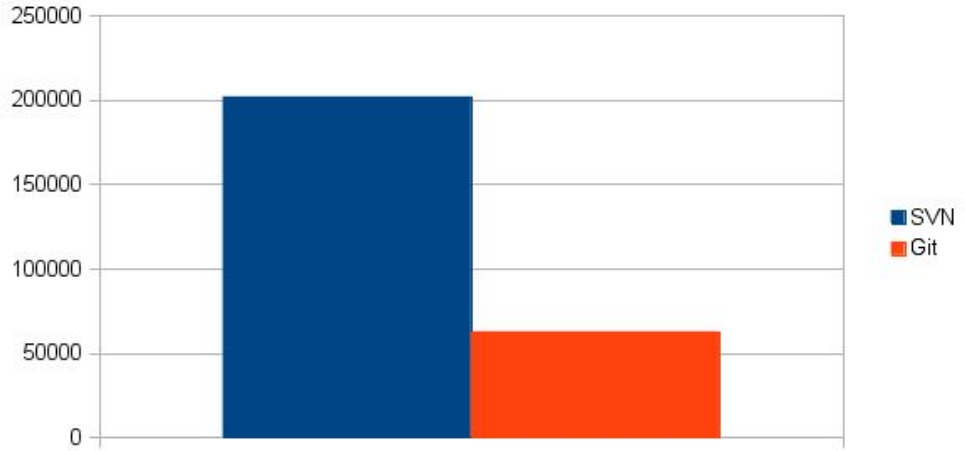
http://www.lourdas.name/blog/git-vs-subversion-performance-comparison
http://vcs.atspace.co.uk/2012/11/05/which-repository-is-more-compact-git-or-svn/



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

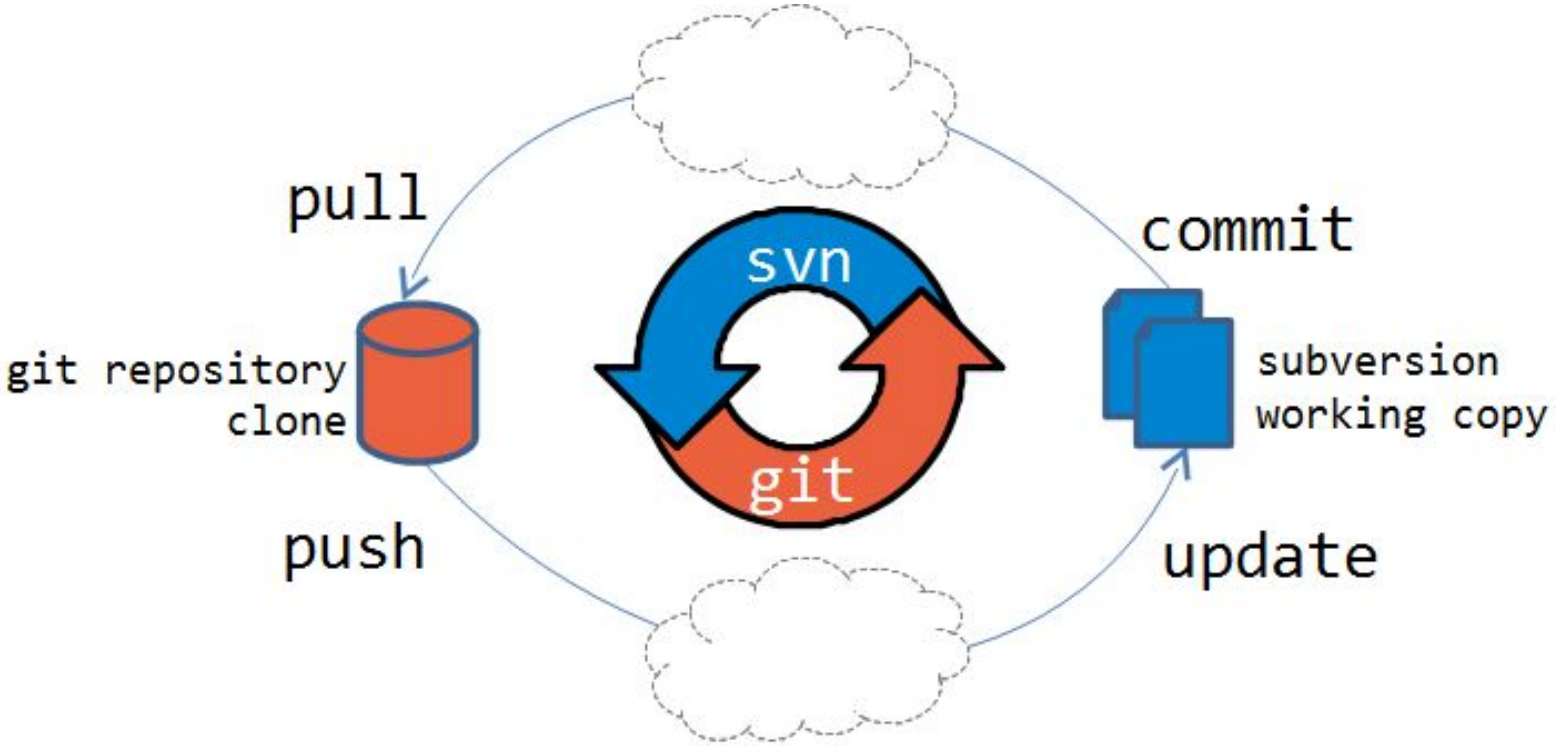
http://www.lourdas.name/blog/git-vs-subversion-performance-comparison
http://vcs.atspace.co.uk/2012/11/05/which-repository-is-more-compact-git-or-svn/

Arggghhh 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. **IU** de SVN mejor...
3. en GIT las tareas simples necesitas muchos **más pasos/comandos** que con SVN.
4. SVN repositorio **único** y **central…**

Instalación y configuración

...OSX ya viene por defecto

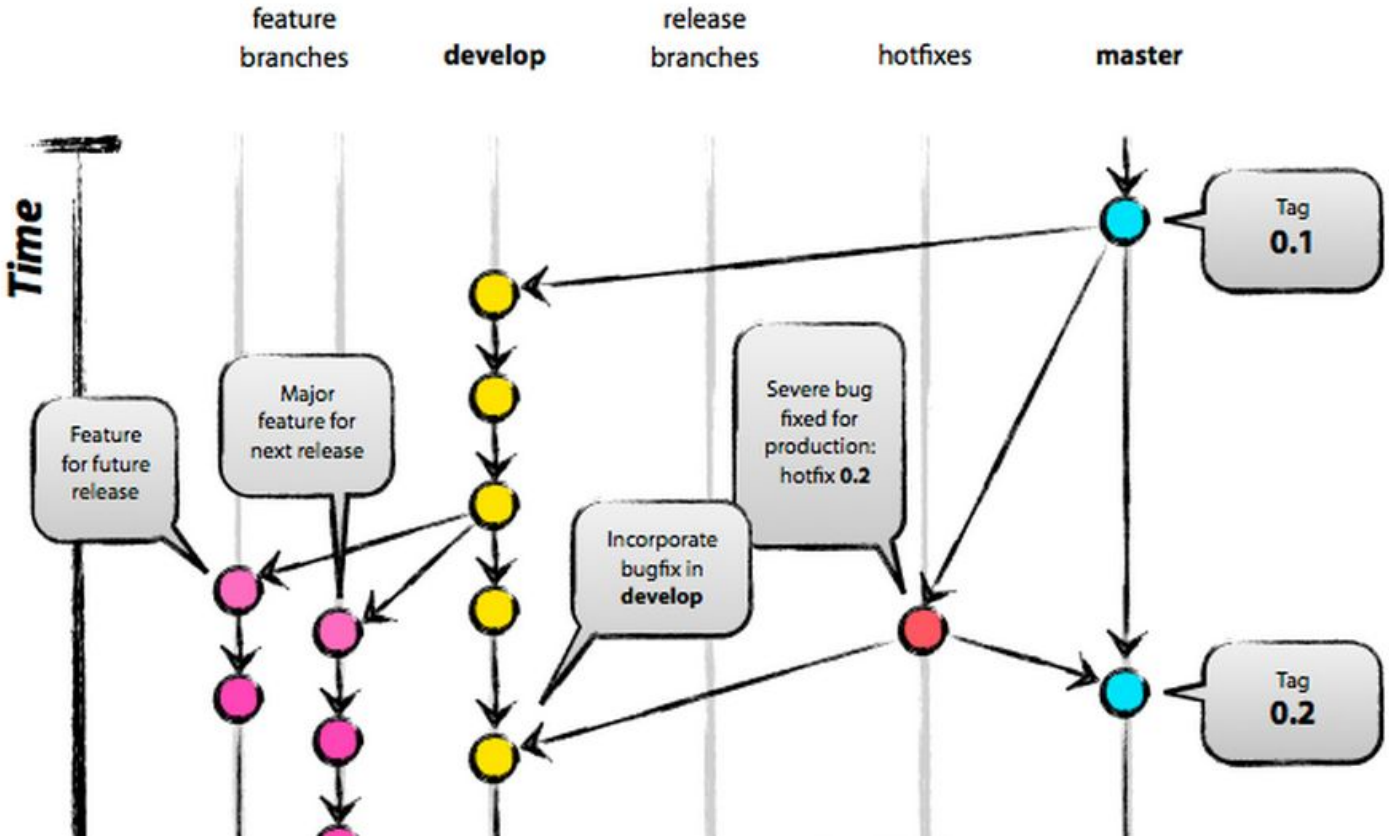
… Ubuntu, apt-get install git-core git-doc

...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!

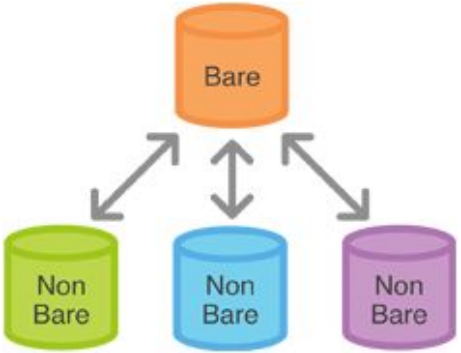

feature branches
develop
release branches
hotfixes
master
Time
Feature for future release
Major feature for next release
Incorporate bugfix in develop
Severe bug fixed for production: hotfix 0.2
Tag 0.1
Tag 0.2

basics: init / clone / .gitignore

$ git init
Inicializa el directorio actual

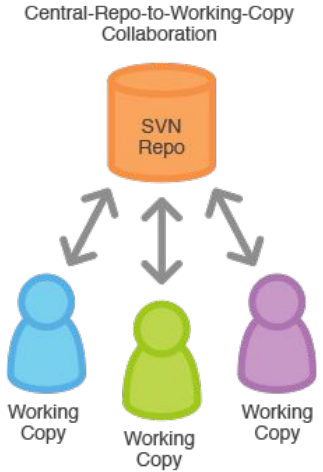
$ git init <directory>
Inicializa el directorio objetivo

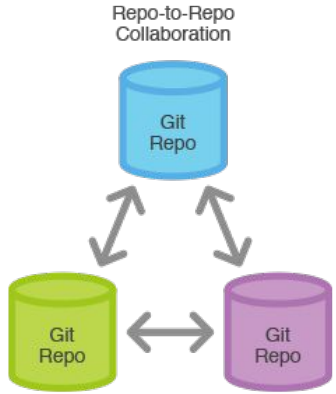
$ git init --bare <directory>
lo inicializa como “bare”



$ git clone <repo>
clona el repo con mismo nombre

$ git clone <repo> <directory>
clona repo con otro nombre local





basics: status / add / commit

STATUS

```
pipboy:nodeexamples luis$ 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

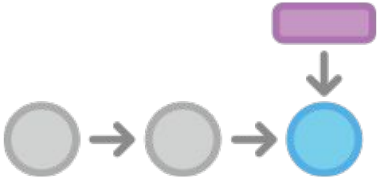
$ git diff <commit> <commit>

branching: branch / merge / checkout

BRANCH

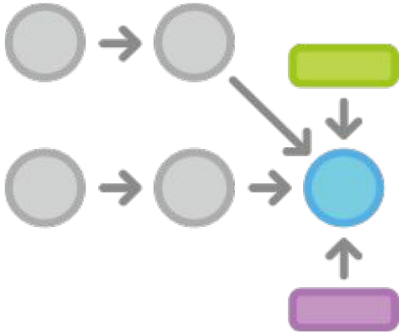
$ git branch

$ git branch <branchname>



MERGE

$ git merge <branch>

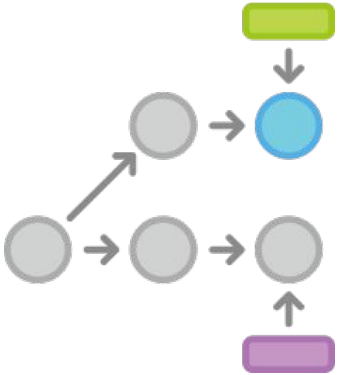


CHECKOUT

$ git checkout <existing-branch>

$ git checkout -b <new-branch>

$ git checkout -b <new> <existing>



branching: tag

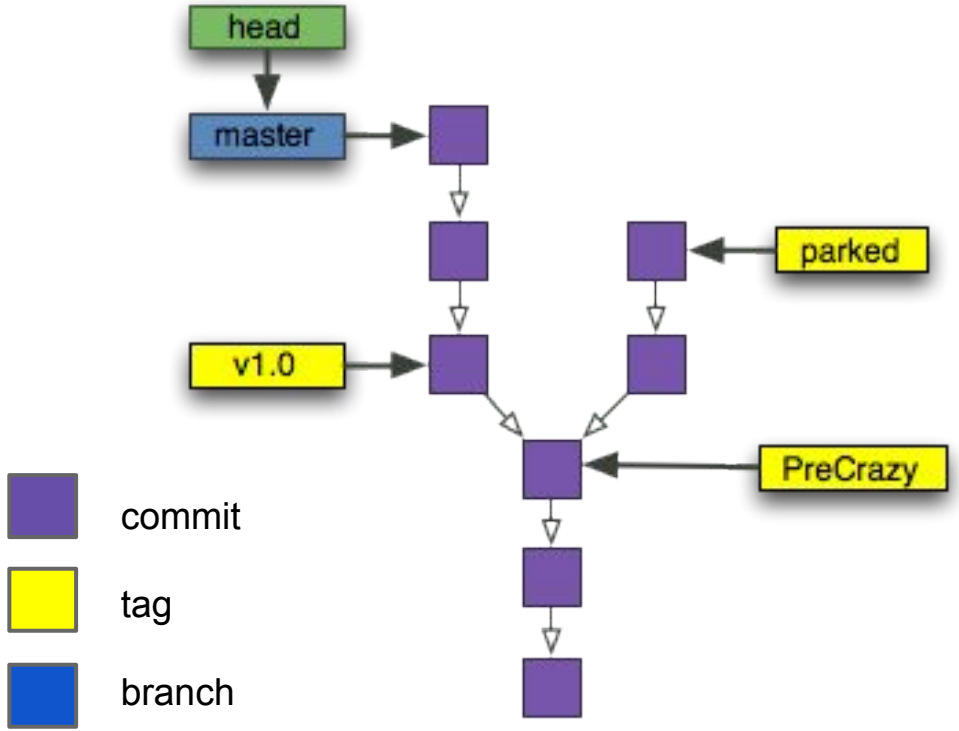
$ git tag
lista

$ git tag v1.4
referencia a un branch

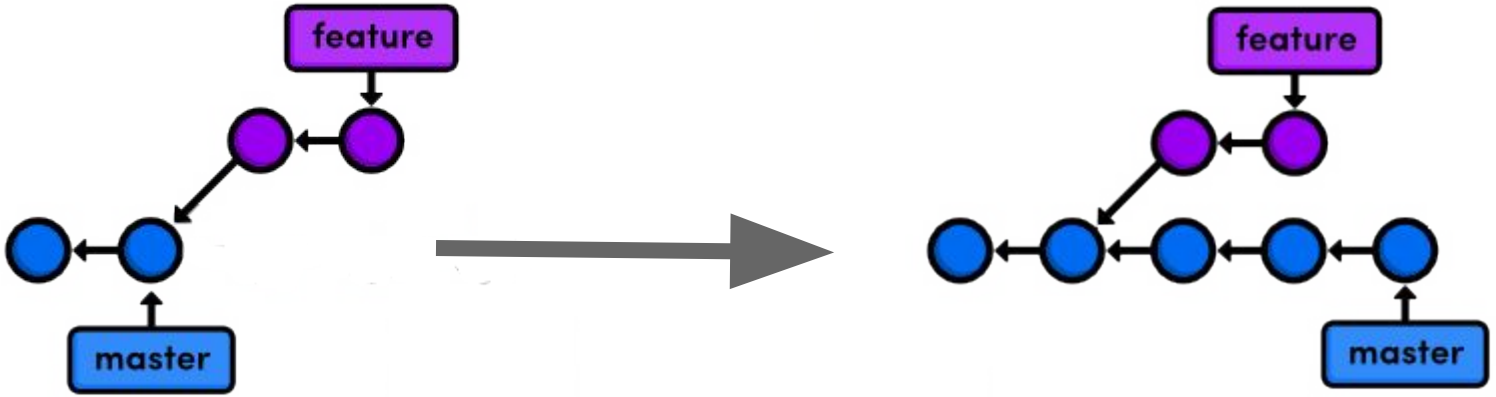
$ git tag -a v1.4 -m 'version 1.4'
checksum, objetos completos

$ git tag -s v1.4 -m 'version 1.4'
firmado con GPG

$ git show v1.4
ver un tag

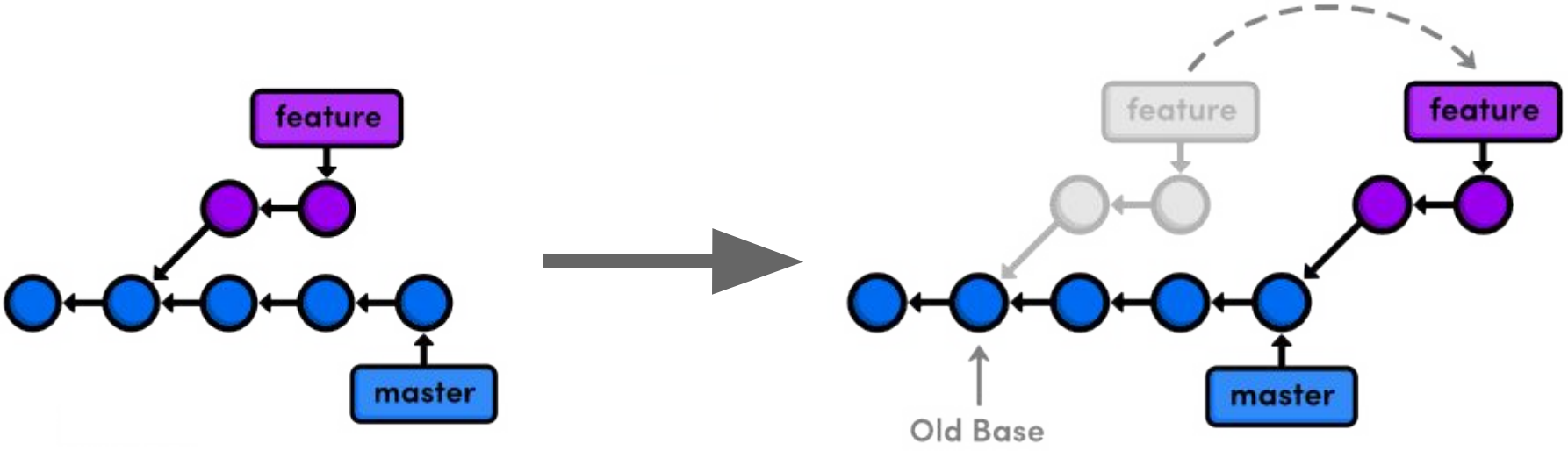


branching: rebase



antes de rebase

branching: rebase

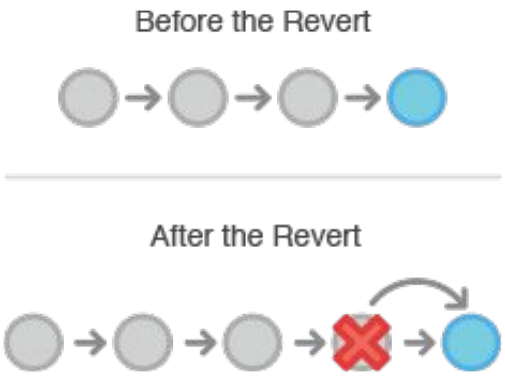


después de rebase

undoing changes...

REVERT

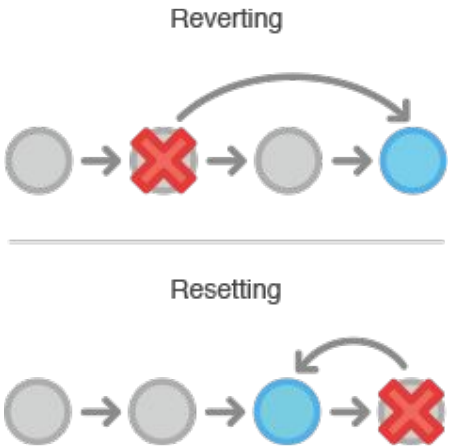
$ git revert <commit>



RESET

$ git reset

$ git reset <file>



CLEAN

$ git clean

distributed: origin / master / remotes

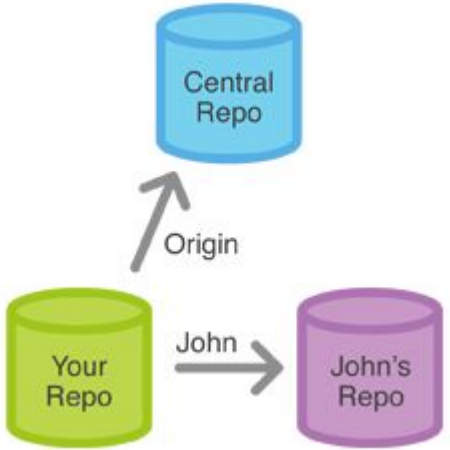
Master es la rama principal de todo repositorio git (equivalente al **trunk**).

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

$git remote add <remotename> <uri>

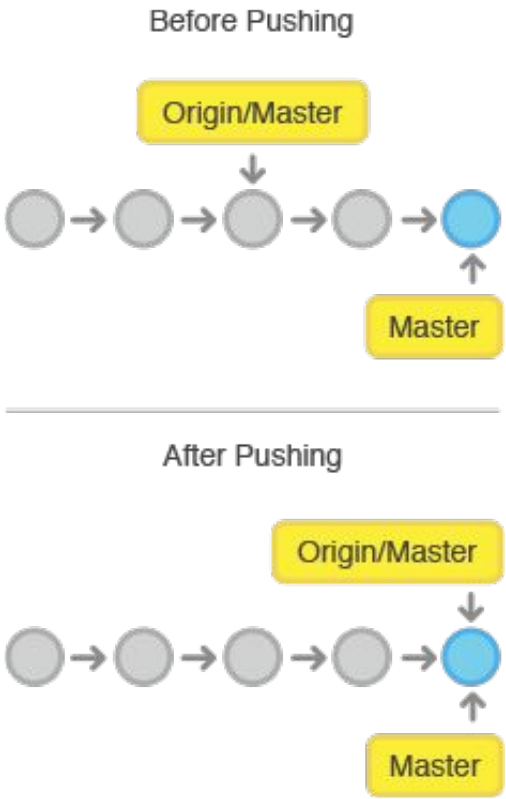
$git remote rm <remotename>

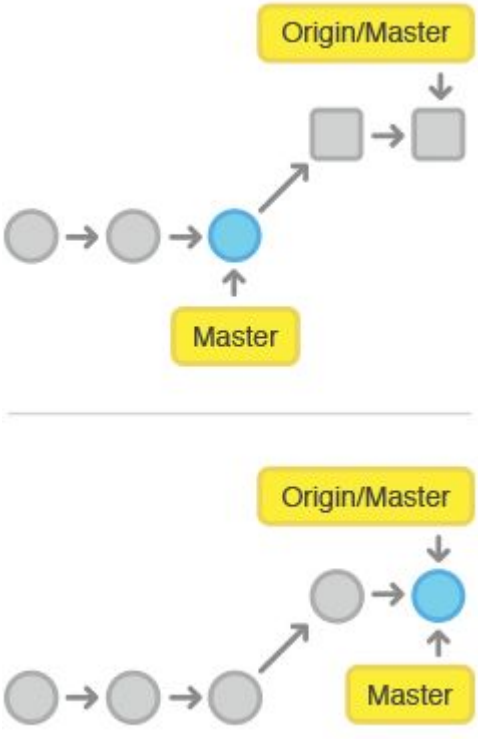
$git remote mv <remotename> <newname>



distributed

$ git push/pull <remote> <branch>

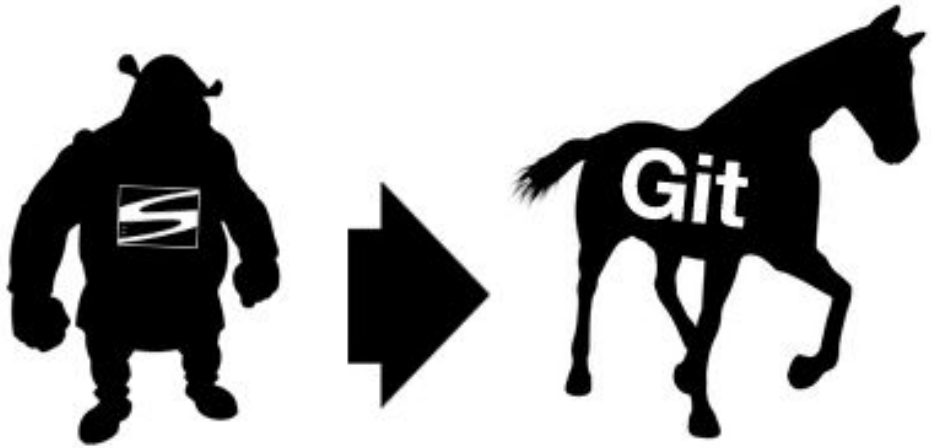




advanced: git-svn

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
```

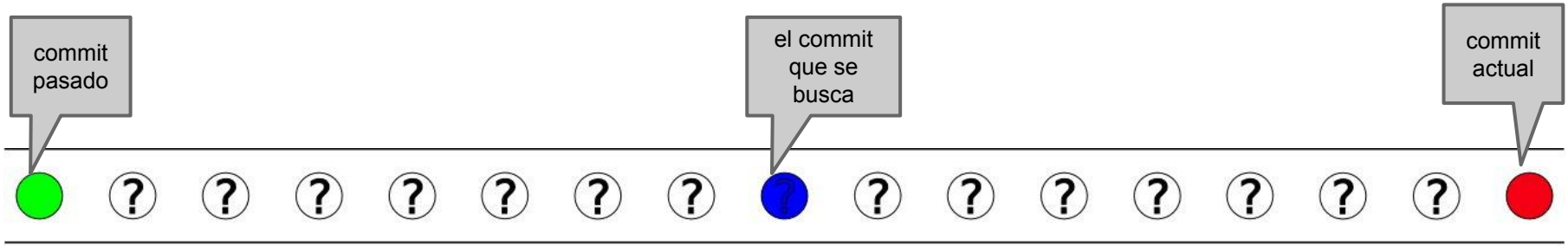
advanced: stash

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.

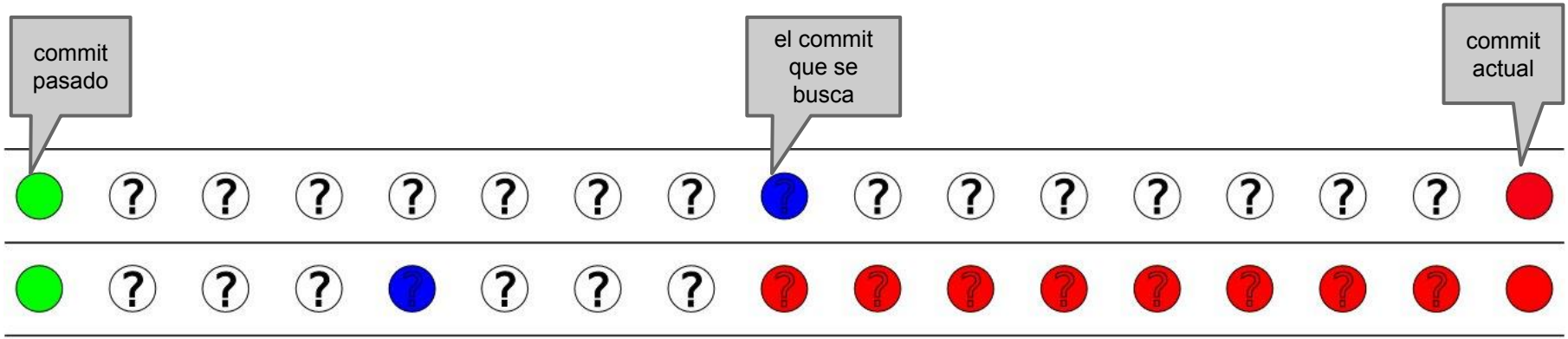
advanced: bisect

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



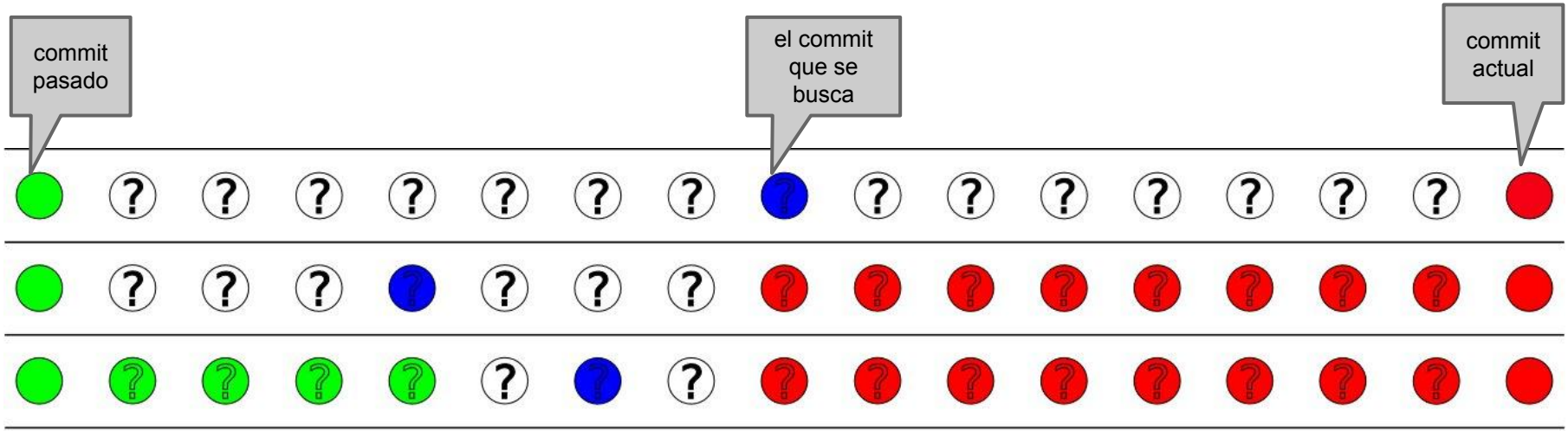
advanced: bisect

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



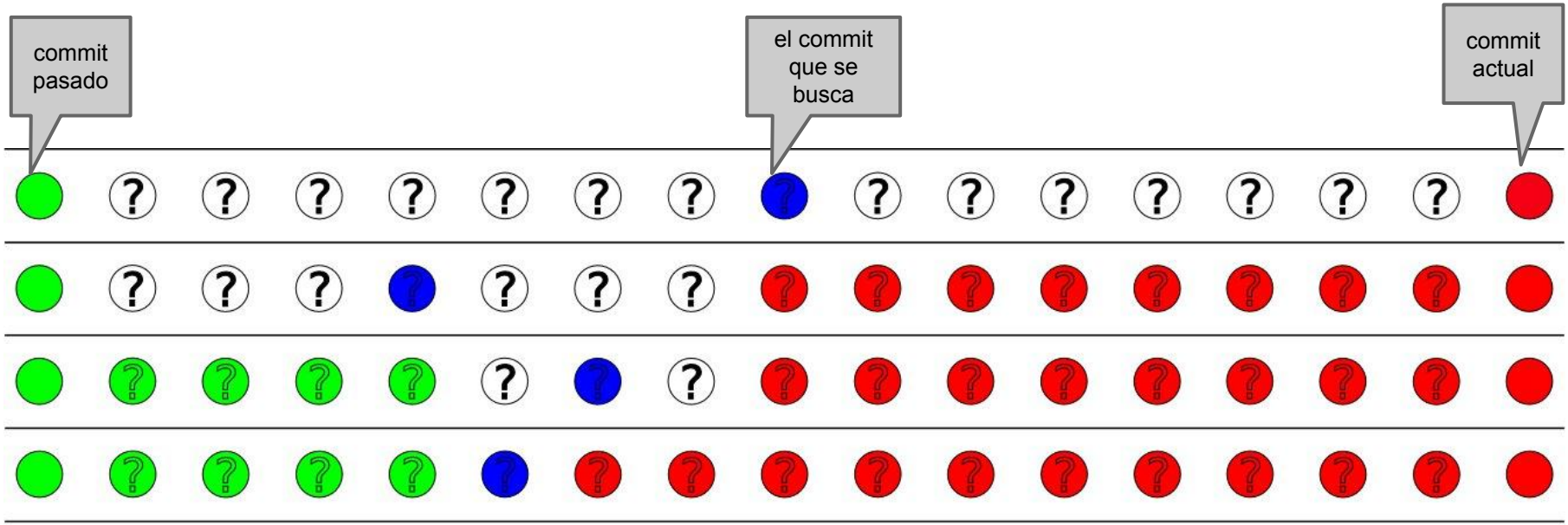
advanced: bisect

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



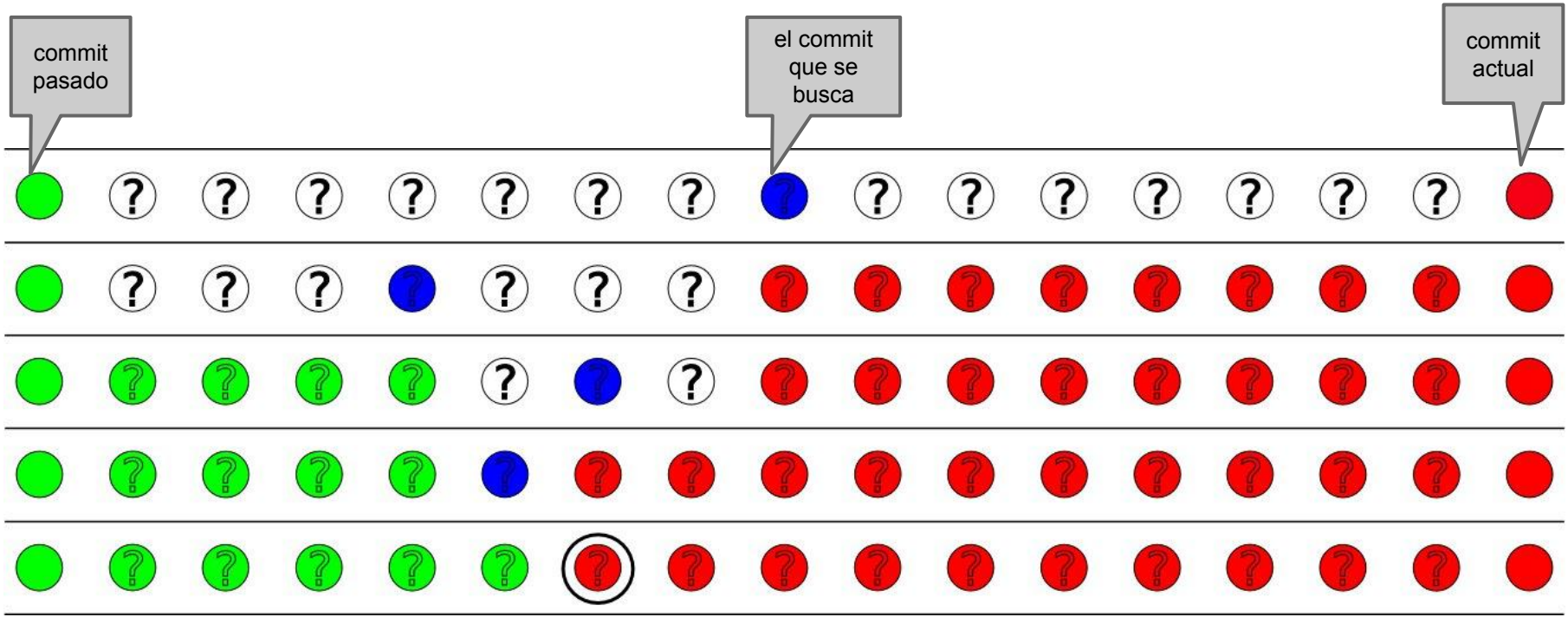
advanced: bisect

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

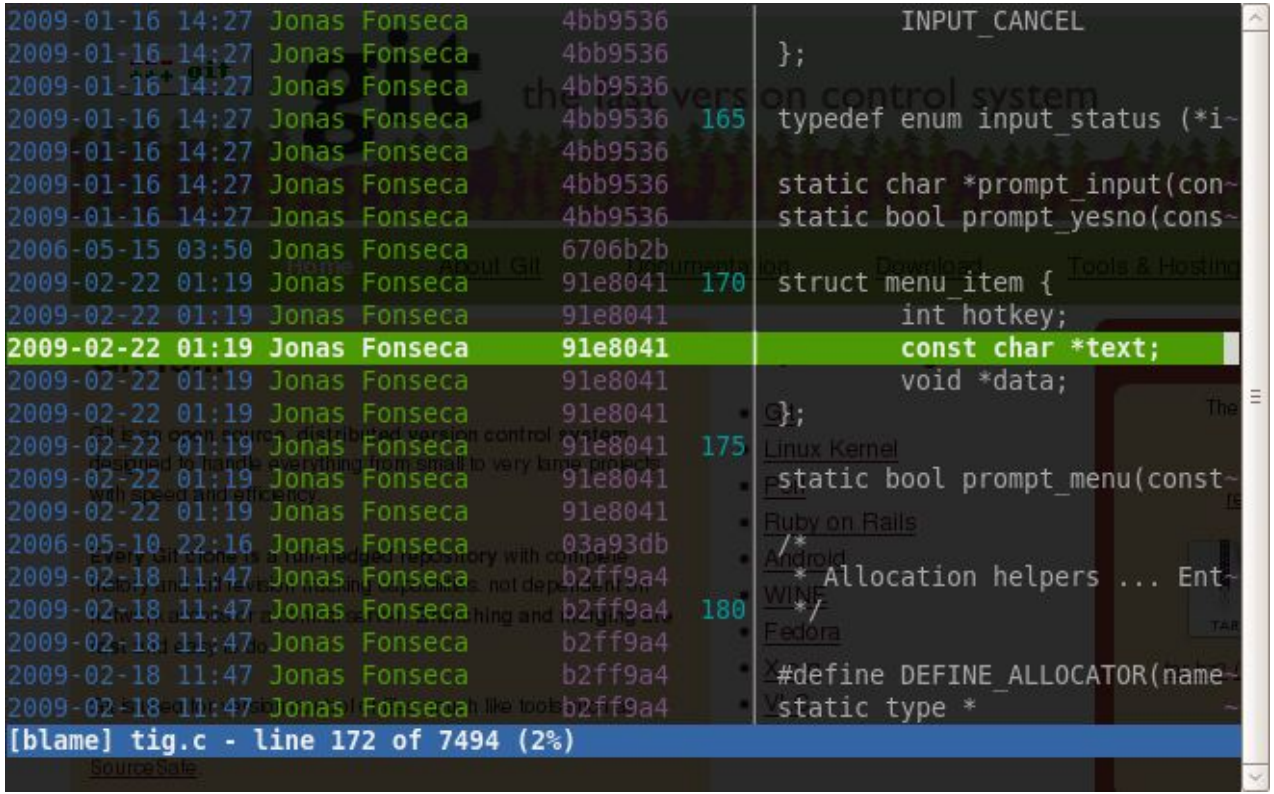


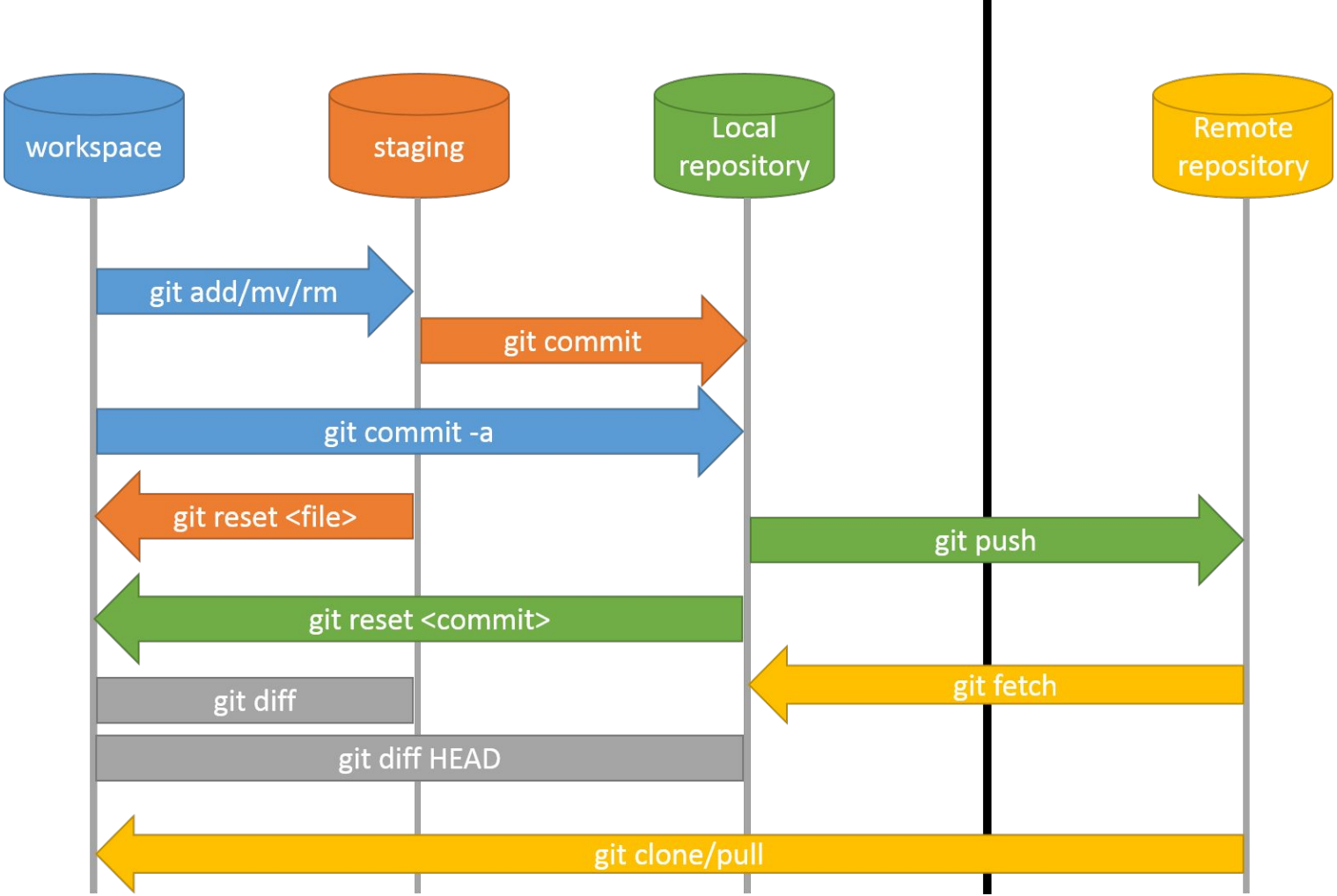
advanced: bisect

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



advanced: blame




workspace
staging
Local repository
Remote repository
git add/mv/rm
git commit
git commit -a
git reset <file>
git push
git reset <commit>
git fetch
git diff
git diff HEAD
git clone/pull

BEEVA

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@beeva.com

   $ git config --list
3. generar rsa (no se tiene que generar si ya tienes una, para saber si la tienes “$ ls -la ~/.ssh”

   $ ssh-keygen -t rsa
4. crearse cuenta en github con tú correo
5. subir la llave pública (.ssh/id_rsa.pub) a nuestra cuenta de github en settings.

   https://github.com/settings/ssh
6. dime tu usuario para que te agregue permisos