

# Gestión de proyectos software con Git y Github



# GIT

## 1. Proyecto, directorio y versión

# GIT

## ◆ GIT: gestor de proyectos software

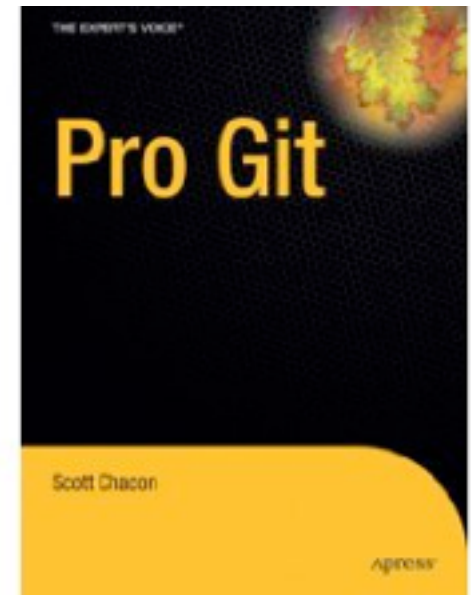
- Desarrollado por Linus Torwalds para Linux

## ◆ Diseñado para desarrollo distribuido

- Cada desarrollador trabaja de forma independiente en su propio repositorio
  - Sincroniza el repositorio con otro cuando necesita
- Uno de los repositorios puede utilizarse como repositorio de referencia

## ◆ Tutorial Web y eBook

- <http://git-scm.com/book/es>

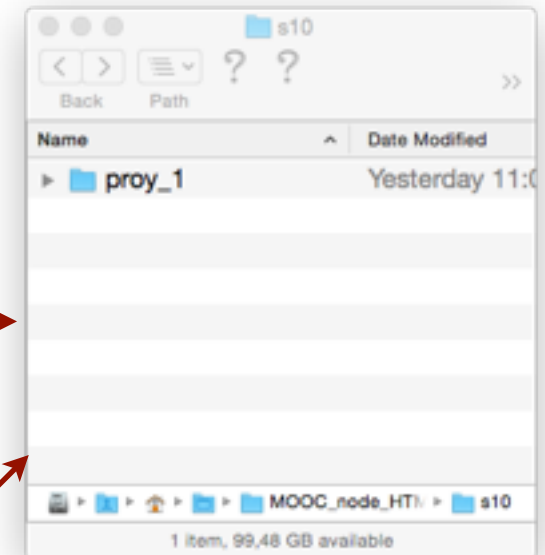


# El directorio del proyecto

- ◆ Un **proyecto** se suele gestionar en un **directorio** (o carpeta)
  - El directorio contiene todos los ficheros del proyecto
- ◆ **Explorador de ficheros**: muestra el contenido de un directorio gráficamente
  - Hacer clic sobre un objetos gráfico ejecuta un comando predefinido
- ◆ **Terminal de comandos**: ejecuta comandos en **directorio de trabajo**
  - El **directorio de trabajo** asociado es la base de las rutas (paths) relativas
    - Los objetos se identifican con rutas (paths) absolutas o relativas

```
venus:s jq$ pwd
/Users/jq/Desktop/MOOC_node_HTML5/s10/s
venus:s jq$ ls
venus:s jq$ ls -a
.  ..
venus:s jq$ mkdir proy_1
venus:s jq$ ls
proy_1
venus:s jq$ ls -a
.  ..      proy_1
venus:s jq$ cd proy_1/
venus:proy_1 jq$ pwd
/Users/jq/Desktop/MOOC_node_HTML5/s10/s/proy_1
venus:proy_1 jq$
```

Terminal de comandos

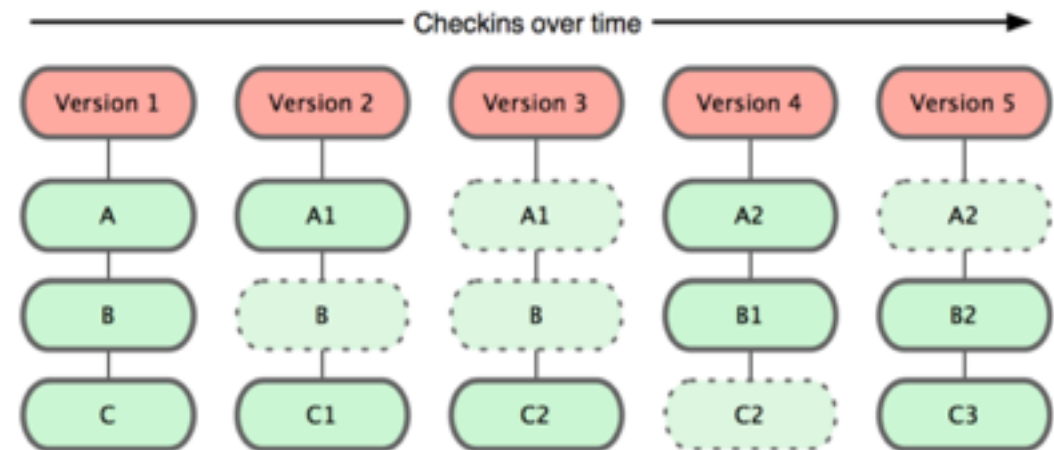
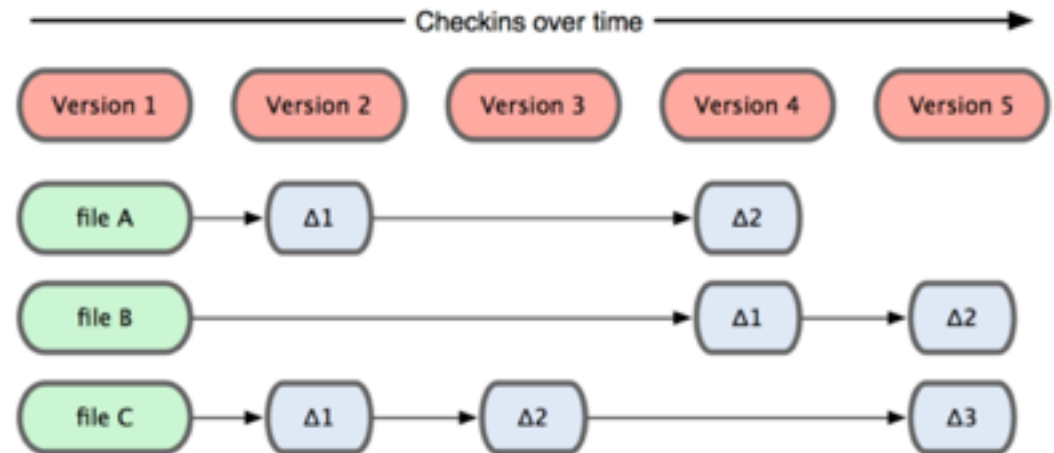


Explorador de ficheros



# Historia de un proyecto

- ◆ Historia de un proyecto:
  - ▶ Historia de cambios en el directorio del proyecto
- ◆ Versión (Commit)
  - ▶ Punto de la historia del proyecto que puede ser restaurado (reconstruido)
- ◆ Se debe consolidar versión en los puntos del desarrollo que deseemos poder volver atrás en el futuro
  - ▶ Versiones frecuentes facilitan el mantenimiento y la legibilidad de un programa



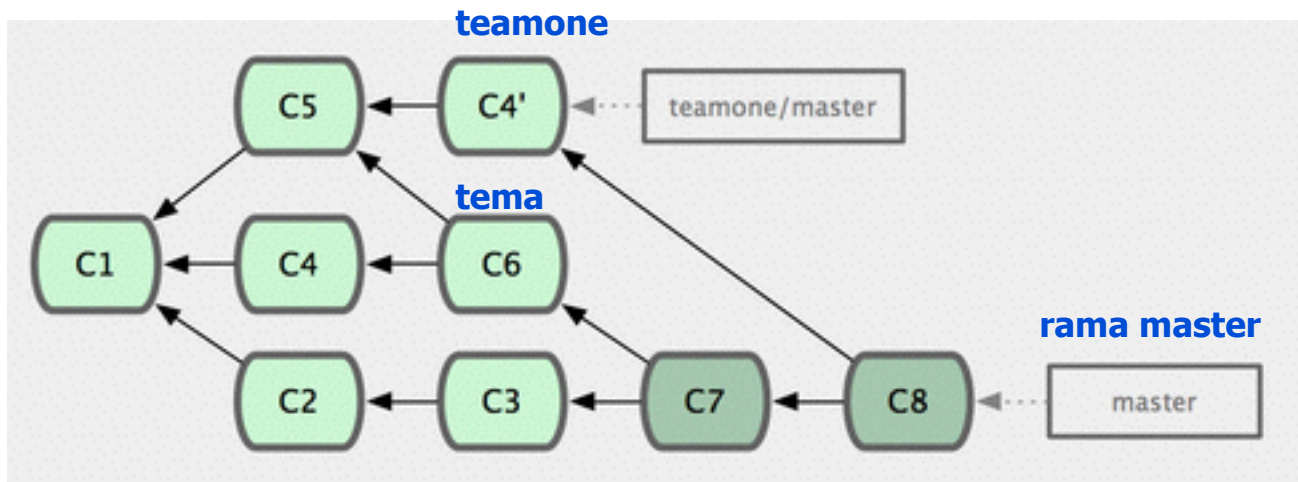
# Árbol de versiones

## ◆ Proyectos software son complejos

- Suelen generar un árbol de versiones

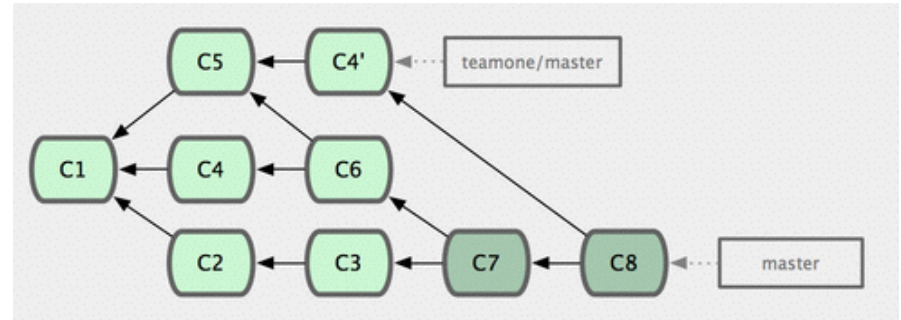
## ◆ La **rama principal** del proyecto se denomina **master**

- En este árbol hay además 2 ramas: **tema** y **teamone**
  - Una rama suele realizar un desarrollo separado
- **Las ramas se suelen integrar en master**, una vez acabadas
  - P. e., la integración puede realizarse con el comando: **git merge teamone**



\*de Scott Chanson: <http://progit.org/book/>

# Repositorio y versión



\*de Scott Chanson: <http://progit.org/book/>

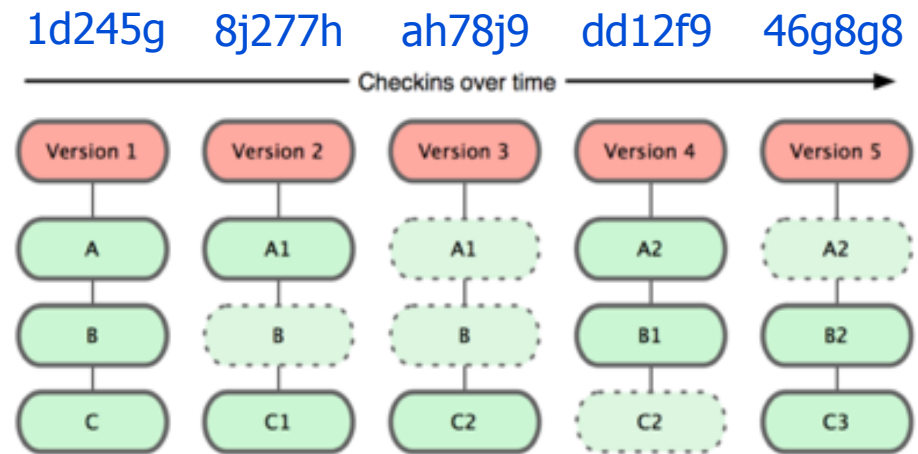
◆ Un **repositorio git** es un “directorio donde gestionar versiones”

- El comando “**git init**” invocado en el directorio
  - Habilita el directorio como un **repositorio git**
  - Puede **guardar o restaurar versiones**
- Las versiones se guardan en el directorio oculto **.git**

◆ **Versión (commit)**

- Directorio (proyecto) congelado en un momento dado
  - Incluyendo todos sus ficheros y subdirectorios
- Punto de sincronización del proyecto que puede restaurarse

# Identificador de versión (SHA1)



◆ Cada versión generada por GIT se identifica con

- Número aleatorio único (clave SHA1)
  - ejemplo: **973751d21c4a71f13a2e729ccf77f3a960885682**

◆ GIT permite equipos de desarrollo distribuidos

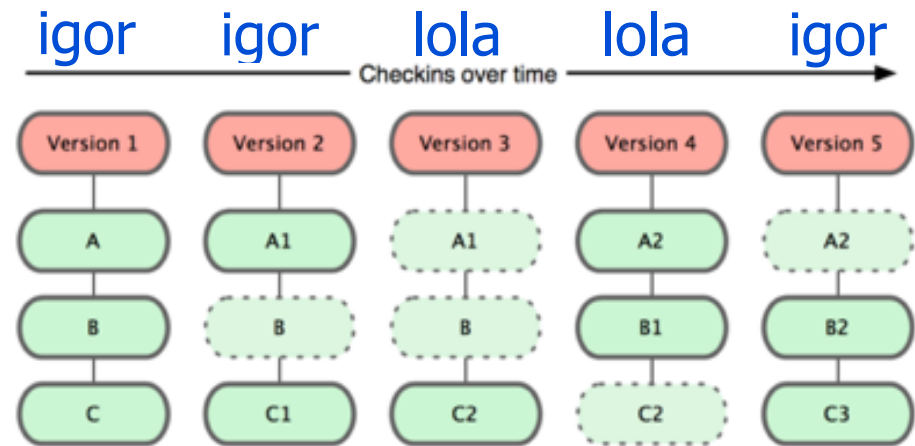
- Los repositorios se pueden clonar sin problemas
  - Ninguna versión en ningún otro repositorio utilizará el mismo identificador

◆ El identificador es muy largo y se suelen utilizar

- los 7-8 dígitos iniciales (únicos en un proyecto): **973751d2**
  - Comandos GIT: permiten **identificadores cortos o largos**



# Colaboración y Firma



\*de Scott Chanson: <http://git-scm.com/book>

- ◆ GIT esta pensado para trabajar en grupo
  - Toda operación va firmada por su autor
    - Al configurar GIT se da el nombre e email del autor
- ◆ Un usuario puede copiar o clonar otro repositorio
  - Y continuar el desarrollo por su cuenta sobre la copia
- ◆ Dos repositorios pueden volver a sincronizarse
  - Aunque integrar las nuevas versiones puede ser complejo

# Configurar GIT

# El comando **"git config"** permite manejar opciones de configuración.

# Las opciones configuradas pueden afectar a distintos ámbitos (proyectos):

# - Para todos los proyectos en el sistema.

# Usar opción **--system**. La configuración se guarda en **/etc/gitconfig**

# - Para todos los proyectos del usuario.

# Usar opción **--global**. La configuración se guarda en **~/.gitconfig**

# - Sólo para el proyecto actual.

# Sin opción. La configuración se guarda en **.git/config**

# Consultar todas las opciones existentes: **git help config**

# Para firmar correctamente contribuciones y versiones debemos configurar:

```
$ git config --global user.name "Pedro Ramirez"
```

```
$ git config --global user.email pramirez@dit.upm.es
```

# Consultar el valor de todas las opciones configuradas:

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

```
user.name=Pedro Ramirez
```

```
user.email=pramirez@dit.upm.es
```

```
color.ui=true
```

# Consultar el valor de una opción:

```
$ git config user.name
```

```
Pedro Ramirez
```

# Ayuda

## # Ayuda en línea de comandos:

\$ git help	# Muestra lista con los comandos existentes
\$ git help comando	# Ayuda sobre comando especificado
\$ git help add	# Ayuda sobre el comando add
\$ git add --help	# Equivalente a anterior
\$ man git-add	# Equivalente a anterior

## # Manual de referencia, chuletas, videos, otros enlaces:

<http://git-scm.com/doc>

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

[https://na1.salesforce.com/help/doc/en/salesforce\\_git\\_developer\\_cheatsheet.pdf](https://na1.salesforce.com/help/doc/en/salesforce_git_developer_cheatsheet.pdf)



# GIT

## 2. Proyecto quiz-2015 en GITHUB

# Proyecto Quiz

- ◆ **Versión 1:** Esqueleto del proyecto con express-generator
- ◆ **Versión 2:** Primera página y el favicon
- ◆ **Versión 3:** Primera pregunta
- ◆ **Versión 4:** Vistas parciales y marco
- ◆ **Versión 5:** CSS adaptable a móviles y PCs
- ◆ **Versión 6:** Despliegue en la nube (Heroku)
- ◆ **Versión 7:** La base de datos: sequelize.js y SQLite
- ◆ **Versión 8:** Desplegar en Heroku con Postgres
- ◆ **Versión 9:** Lista de preguntas
- ◆ **Versión 10:** Autoload de la DB
- ◆ **Versión 11:** Crear preguntas
- ◆ **Versión 12:** Validación de entradas
- ◆ **Versión 13:** Editar preguntas
- ◆ **Versión 14:** Borrar preguntas
- ◆ **Versión 15:** Crear comentario
- ◆ **Versión 16:** Autenticación y sesión
- ◆ **Versión 17:** Autorización
- ◆ **Versión 18:** Moderación de comentarios
- ◆ **Versión 19:** HTTPS - HTTP Seguro

**Objetivo:** Crear un pequeño portal Web con un juego de adivinanzas (quizes) usando MVC y vistas adaptables a móvil. Quiz ilustra también el uso de herramientas de gestión de proyectos. El proyecto solo tiene una rama: **master**

El proyecto Quiz en **GITHUB**  
<https://github.com/jquemada/quiz-2015>

Proyecto desplegado y operativo en **heroku**  
<https://quiz-2015.herokuapp.com/>



GitHub repository page for **jquemada / quiz-2015**. The page shows 19 commits, 2 branches, 0 releases, and 2 contributors. The current branch is **quiz-2015**. The file list on the left includes `bin`, `certs`, `controllers`, `models`, `public`, `routes`, `views`, `.gitignore`, `Procfile`, `app.js`, `commands_to_generate_keys.txt`, and `package.json`. The commit history shows a series of updates, including "Soporte HTTPS", "Moderación de comentarios", "Autenticación de usuarios", and "Despliegue en Heroku".

URL que identifica el repositorio:  
<https://github.com/jquemada/quiz-2015>

## Quiz en GITHUB

**branches  
(ramas)**

**Commits  
(versiones)**

**código:  
directorios,  
ficheros, ..**

30 lines (29 sloc) 0.584 kb

```
1 {
2   "name": "quiz",
3   "version": "0.0.0",
4   "private": true,
5   "scripts": {
6     "start": "node ./bin/www"
7   },
8   "engines": {
9     "node": "0.10.x",
10    "npm": "1.4.x"
11  },
12  "devDependencies": {
13    "sqlite3": "^3.0.4"
14  },
15  "dependencies": {
16    "body-parser": "~1.8.1",
17    "cookie-parser": "~1.3.3",
18    "debug": "~2.0.0",
19    "ejs": "~0.8.5",
20    "express": "~4.9.0",
21    "express-partials": "^0.3.0",
22    "express-session": "~1.0.3",
23    "method-override": "^2.3.1",
24    "morgan": "~1.3.0",
25    "pg": "^4.1.1",
26    "sequelize": "^2.0.0-rc4",
27    "serve-favicon": "~2.1.3"
28  }
29 }
```

# Commits (versiones)

Último commit (versión)

identificador  
"corto" de commit

Diferencias con la versión anterior  
del fichero views/index.ejs:  
**rojo:** eliminado, **verde:** añadido

commits  
anteriores:  
historia de  
versiones

Commits · jquemada/quiz-2015 · Settings | Heroku | Application Error

GitHub, Inc. (US) | https://github.com/jquemada/quiz-2015/commits/master

Bancos | Varios | Cursos | Tools | Rails | HTML5 | HTML5-RTC | MITwitter | CTING | DIT | FloorControl | Most Visited | Global3

GitHub | This repository | Search | Explore | Features | Enterprise | Blog | Sign up | Sign in

jquemada / quiz-2015 | Watch 1 | Star 0 | Fork 1

branch: master

Commits on Mar 1, 2015

- Soporte HTTPS  
jqemada authored 10 days ago

Commits on Feb 27, 2015

- Moderación de comentarios  
jqemada authored 10 days ago

Commits on Feb 26, 2015

- Autorización  
jqemada authored 10 days ago
- Autenticación de usuarios  
jqemada authored 11 days ago

Commits on Feb 6, 2015

- Crear comentario  
ebarra authored on 6 Feb

Commits on Feb 2, 2015

343a8ef

4 views/index.ejs

5	5	<link rel='stylesheet' href='/stylesheets/style.	
6	6	</head>	
7	7	<body>	
8		- <h1><%= title %></h1>	
9		- <p>Welcome to <%= title %></p>	
	8	+ <h1>Bienvenido a <%= title %></h1>	
	9	+ <p>El portal donde podrá crear sus propios juegos	
10	10	</body>	
11	11	</html>	

# Clonar e inspeccionar quiz-2015

```
# Un repositorio público en GITHUB, o en otro servidor al que tengamos acceso,  
# puede clonarse en nuestro ordenador con: git clone <URL_repositorio>  
# -> la copia incluye el proyecto completo con toda su historia de versiones  
  
# Podemos clonar el proyecto quiz-2015 de la copia en GITHUB con el comando:  
$ git clone https://github.com/jquemada/quiz-2015  
# El proyecto se copia en un nuevo directorio llamado quiz-2015  
  
# También podemos indicar cual es el nombre del directorio a crear  
$ git clone https://github.com/jquemada/quiz-2015 mi_proyecto  
# El proyecto se copia ahora en un nuevo directorio llamado mi_proyecto  
  
$ cd quiz-2015 # Entramos en el directorio clonado quiz-2015 (o mi-proyecto)  
  
$ git log --oneline # Muestra las versiones del proyecto  
  
$ git checkout <commit_id_SHA1> # descongela las versiones de la historia  
$ git checkout master # vuelve a la rama (última versión: Quiz 19)  
$
```



# GITHUB App for MAC & for Windows I

**GITHUB App** es una herramienta gráfica muy eficaz para gestionar proyectos git localmente en el PC. Solo está soportada para MAC y para Windows:

Descargar (e instalar) GITHUB for MAC: <https://mac.github.com>

Descargar (e instalar) GITHUB for Windows: <https://windows.github.com>

El proyecto Quiz de GITHUB (<https://github.com/jquemada/quiz-2015>), una vez clonado en un directorio local, puede añadirse a GITHUB App tal y como se indica en la figura.

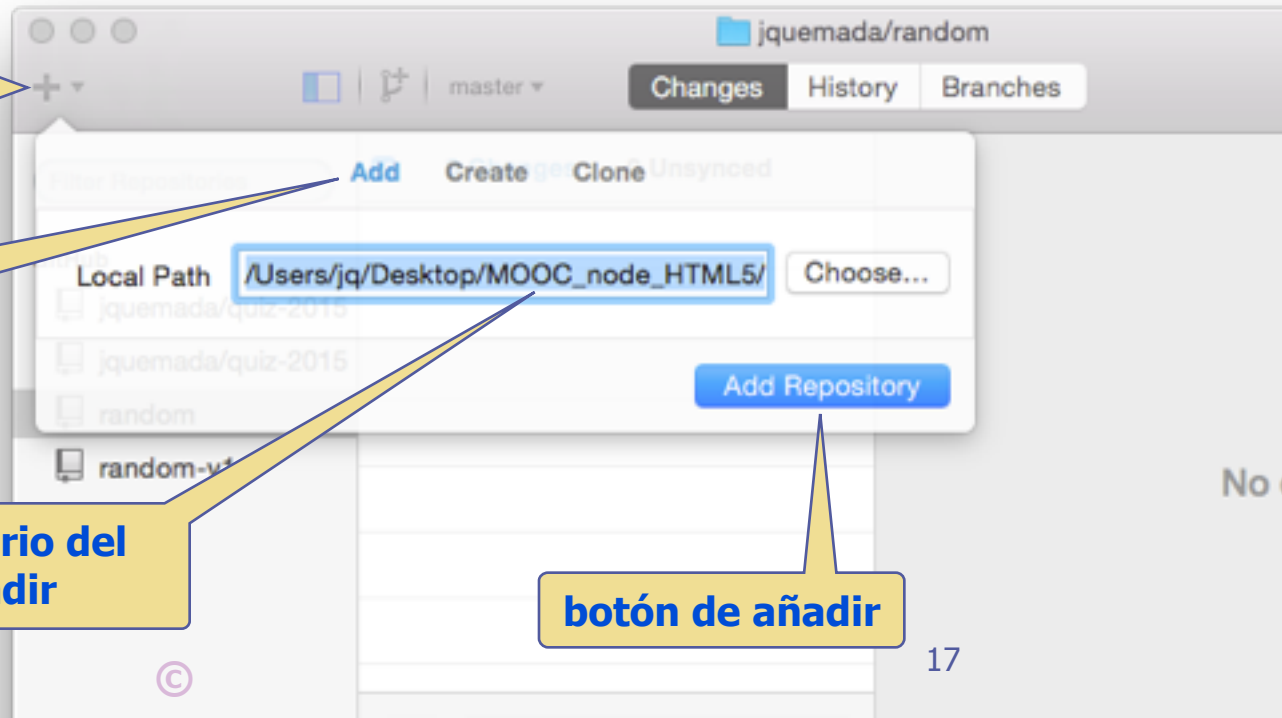
**Desplegable para:**

- **Añadir (Add) proyecto**
- **Crear (Create) proyecto**
- **Clonar (Clone) proyecto**

**Pestaña de  
añadir proyecto**

**ruta al directorio del  
proyecto a añadir**

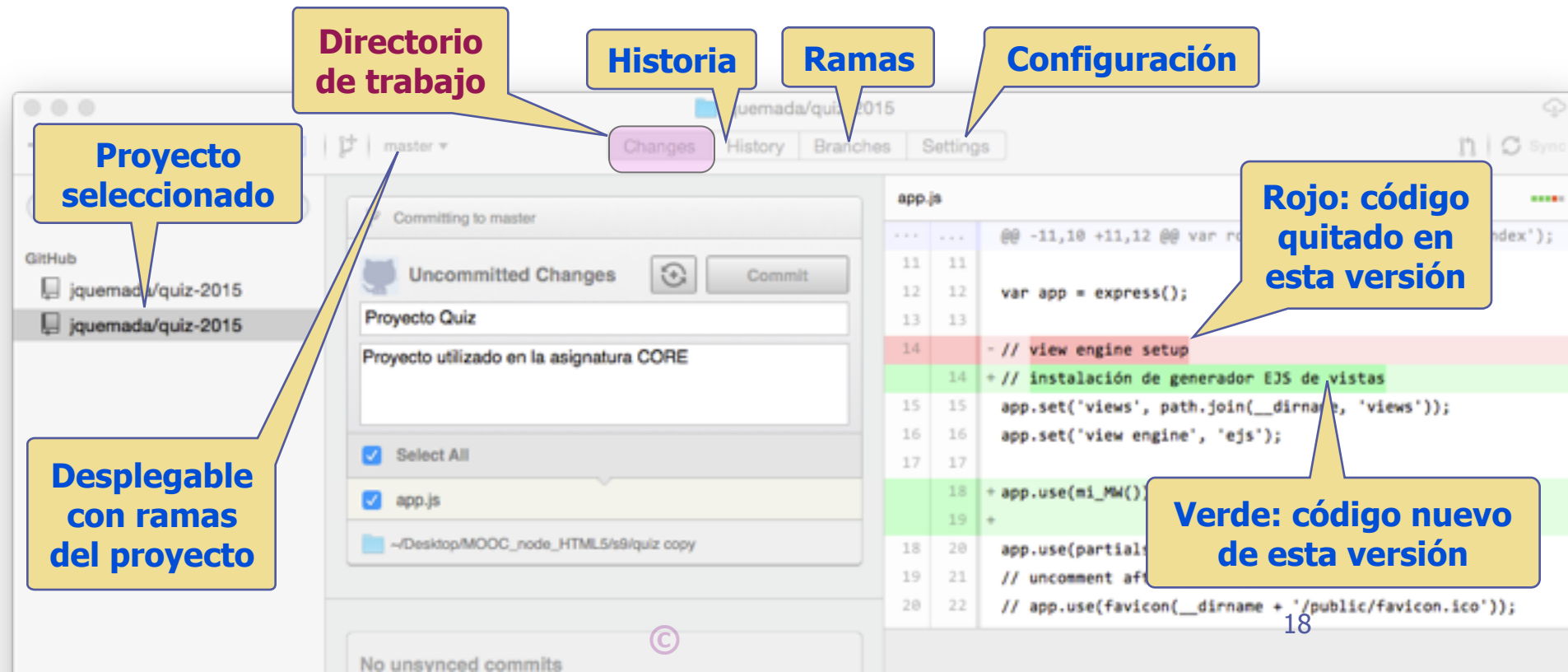
**botón de añadir**



# GITHUB App for MAC & for Windows II

GITHUB App for MAC/Windows **gestiona proyectos git localmente en nuestro PC**. Permite

- Ver y gestionar el **directorio de trabajo** del proyecto y los **cambios realizados**.
- Ver las **versiones de un proyecto (historia)** y los **cambios** realizados en cada versión.
- Ver y gestionar las **ramas de un proyecto** y su sincronización con **repositorios remotos**.



# GITHUB App for MAC & Windows III

Seleccionando la historia de una rama del proyecto aparecen todas las versiones (commits) de dicha rama.

Seleccionando una versión podemos ver todas las diferencias con la versión anterior.

El código añadido se resalta en verde y el código eliminado se resalta en rojo.

The screenshot displays the GitHub web interface for a repository named 'jqemada/quiz-2015'. The top navigation bar includes 'Changes', 'History', 'Branches', and 'Settings'. The 'History' tab is selected, showing a list of 19 commits. The commit list on the left includes entries like 'Crear comentario', 'Borrar pregunta', 'Editar pregunta', 'Validación', 'Crear pregunta', 'Autoload', 'Lista de Preguntas', 'Despliegue DB en Heroku', 'Modelo y base de datos', 'Despliegue en Heroku', 'Diseño responsivo', 'express-partials', 'Primera pregunta', 'Primera página y favicon', and 'esqueleto express-generator'. The 'Lista de Preguntas' commit is selected, showing a diff view. The diff view compares the selected commit with the previous one. The code is color-coded: red for removed code and green for added code. Callouts highlight these features: 'Historia' points to the 'History' tab, 'Versión' points to the selected commit, 'Rojo: código quitado en esta versión' points to the red lines in the diff, and 'Verde: código nuevo de esta versión' points to the green lines in the diff.

**Historia**

**Versión**

**Rojo: código quitado en esta versión**

**Verde: código nuevo de esta versión**

```
... -4,8 +4,6 @@ var app = require('../app');
4 4
5 5 app.set('port', process.env.PORT || 3000);
6 6
7 - console.log("Llega aqui 1")
8 7 var server = app.listen(app.get('port'), function() {
9 8   debug('Express server listening on port ' + server.address().port);
10 9 });
11 - console.log("Llega aqui 2")

controllers/quiz_controller.js
... -1,19 +1,28 @@
1 1 var models = require('../models/models');
2 2
3 - // GET /quizes/question
4 - exports.question = function(req, res) {
5 -   models.Quiz.findAll().then(function(quiz) {
6 -     res.render('quizes/question', { pregunta: quiz[0].pregunta});
7 3 + // GET /quizes
8 4 + exports.index = function(req, res) {
9 5 +   models.Quiz.findAll().then(function(quizes) {
10 6 +     res.render('quizes/index.ejs', { quizes: quizes});
11 7   });
12 8   };
13 9   };
14 10 - // GET /quizes/answer
15 10 + // GET /quizes/:id
16 11 + exports.show = function(req, res) {
17 12 +   models.Quiz.find(req.params.quizId).then(function(quiz) {
18 13 +     res.render('quizes/show', { quiz: quiz});
19 14 +   });
20 15 + };
21 16 +
22 17 + // GET /quizes/:id/answer
23 18 exports.answer = function(req, res) {
```

# GITHUB App for MAC & for Windows IV

**Abrir o crear proyectos**

**Directorio o carpeta de S.O.**

**Proyectos**

- View on GitHub
- Open in Finder
- Open in Terminal
- Open in Atom
- Remove

**Click con Botón der. del ratón (^Click) en proyecto despliega esta ventana de acceso**

**Atom es un editor muy sencillo y eficaz para editar todos los ficheros del proyecto. Si se utiliza GITHUB for MAC/Windows es conveniente instalarlo.**

**Consola de comando en directorio de trabajo del proyecto**

```
gitignore - /Users/jq/Desktop/MOOC_node_HTML5/s...  
www  
├── controllers  
├── .DS_Store  
├── quiz_cont...  
├── models  
├── .DS_Store  
├── models.js  
├── quiz.js  
├── node_modules  
├── public  
├── routes  
├── quizzes  
├── .DS_Store  
├── error.ejs  
├── index.ejs  
├── layout.ejs  
├── .DS_Store  
├── .env  
├── .gitignore  
├── app.js  
└── package.json
```

Committing to master

Uncommitted Changes

Proyecto Quiz

Proyecto utilizado en la asignatura CORE

app.js

```
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

```
app.js  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

```
quiz copy  
Back Path View Arrange Show  
Favorites Name Date Modified  
Devices  
Tags  
Red  
Orange  
Yellow  
Green  
Purple  
Gray  
All Tags...  
app.js Today 07:58  
bin 31 Jan 2015 1  
controllers 1 Feb 2015 08  
models 1 Feb 2015 08  
node_modules 2 Feb 2015 13  
package.json 31 Jan 2015 1  
Procfile 31 Jan 2015 1  
public 29 Dec 2014 1  
favicon.ico 29 Dec 2014 1  
images 25 Nov 2014 1  
javascripts 25 Nov 2014 1  
stylesheets 1 Feb 2015 08  
quiz.sqlite 2 Feb 2015 15  
routes 1 Feb 2015 08  
index.js 1 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 2015 08  
views 2 Feb 2015 14  
error.ejs 25 Nov 2014 1  
index.ejs 31 Jan 2015 1  
layout.ejs 2 Feb 2015 14  
quizzes 2 Feb 2015 14  
_form.ejs 1 Feb 2015 07  
answer.ejs 31 Jan 2015 1  
edit.ejs 1 Feb 2015 12  
index.ejs 2 Feb 2015 15  
new.ejs 1 Feb 2015 07  
show.ejs 1 Feb 
```



# GIT

## 3. GITHUB

# GITHUB

## ◆ Portal de repositorios GIT (<https://github.com>)

- Enfoque social y colaborativo -> “social coding”
  - Red social para compartir proyectos software
- Curso necesita cuenta en GITHUB
  - Alberga proyectos de la asignatura

## ◆ Repositorios **públicos son gratis**, los privados de pago

- Repositorios totales: +20M (Linux, Eclipse, jQuery, RoR, ...)

## ◆ Gestión de organizaciones y proyectos software

- Soporta equipos de desarrollo distribuidos, abiertos o privados
- Uso y acceso muy sencillo a versiones, tareas, bugs, ...
- Herramientas para desktop (MAC y Windows)
- Incluye muy buenos tutoriales
- ....

# GITHUB: registro y ayuda

- # Lo primero es crear una cuenta y una vez creada, debemos seguir sus instrucciones para
- # -> <https://help.github.com/articles/set-up-git/>
- # 1) Configurar y conectar con GITHUB nuestro GIT local
- # 2) Instrucciones para crear y clonar repositorios
- # 3) Instrucciones para colaborar en proyectos software distribuidos

Click aquí: Instrucciones

**GitHub Bootcamp**

- 1 Set up Git**  
A quick guide to help you get started with Git.
- 2 Create repositories**  
Repositories are where you'll work and collaborate on projects.
- 3 Fork repositories**  
Forking creates a new, unique project from an existing one.
- 4 Work together**  
Send pull requests, follow friends. Star and watch projects.



Repositorio: <https://github.com/jquemada/quiz-2015>  
Puede clonarse con:  
`..$ git clone https://github.com/jquemada/quiz-2015`

**seguir proyecto**

**clonar en mi cuenta**

**branches (ramas)**

**releases**

**Commits (versiones)**

**código: directorios, ficheros, ..**

**colaboradores**

**contribuciones**

**clonar o bajar como ZIP**

Proyecto de CORE 2015 y @NodeMOOC — Edit

19 commits 2 branches 0 releases 2 contributors

branch: master quiz-2015 / +

Soporte HTTPS

jquemada author 10 days ago

bin

certs

controllers

models

public

routes

views

.gitignore

Procfile

app.js

commands\_to\_generate\_keys.txt

package.json

Soporte HTTPS 8 days ago

Soporte HTTPS 8 days ago

Moderación de comentarios 10 days ago

Moderación de 10 days ago

Autenticación de usuarios 10 days ago

Moderación de comentarios 10 days ago

Moderación de comentarios 10 days ago

Despliegue DB en Heroku 2 months ago

Despliegue en Heroku ago

Autenticación de usuarios ago

Soporte HTTPS ago

Autenticación de usuarios 10 days ago

Issues 0

Pull Requests 0

Wiki

Pulse

Graphs

HTTPS clone URL

<https://github.com/jquemada/quiz-2015>

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Clone in Desktop

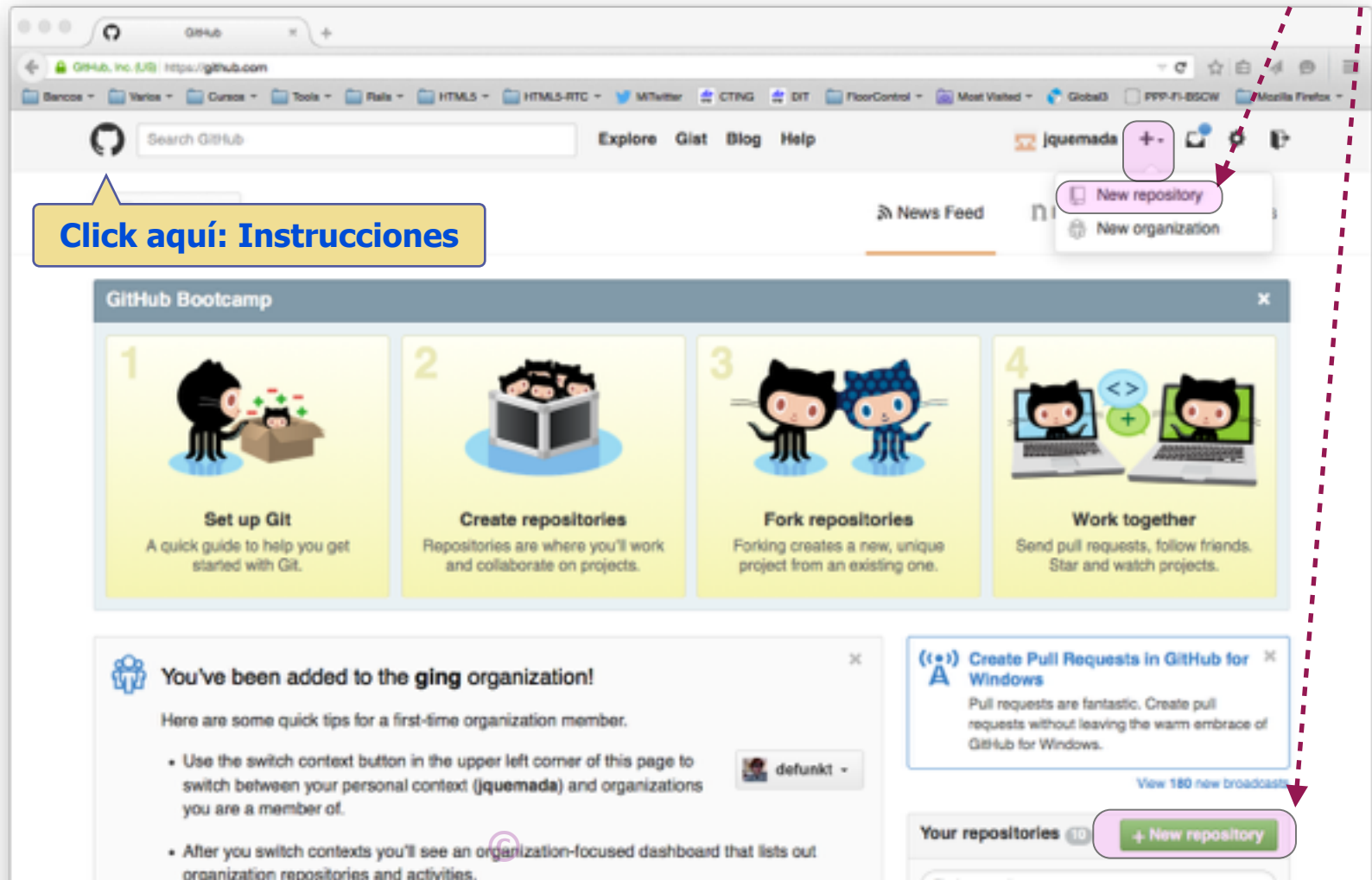
Download ZIP



# Subir un repositorio local a GITHUB

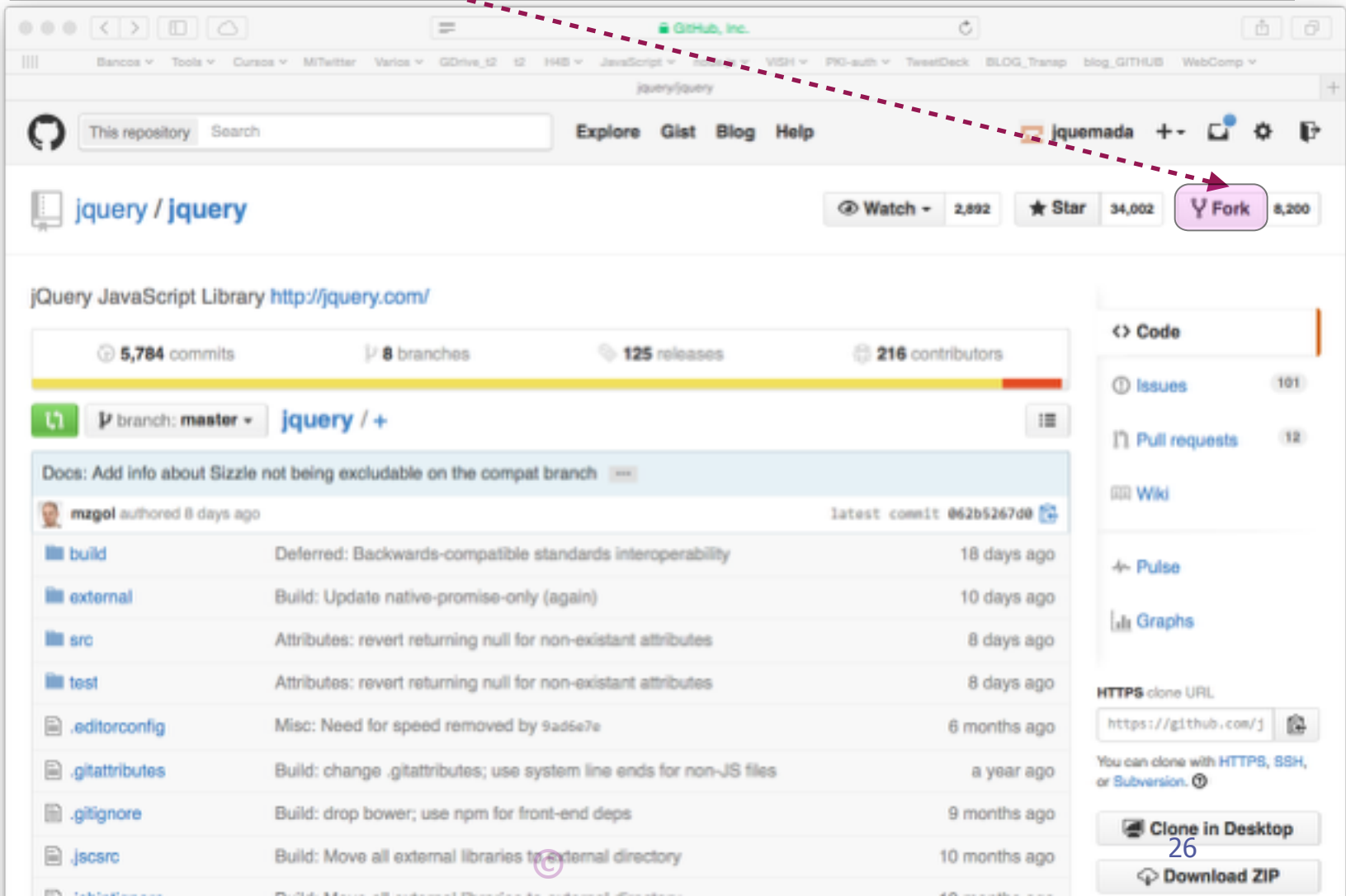
- # Para subir un repositorio local a GITHUB debemos
- # 1) Crear un repositorio vacío en GITHUB con **New Repository**
- # 2) Configurar repo. remoto origin con repositorio vacío
  - \$ `git remote add origin https://github.com/pepe/proy1`
- # 3) Hacer **push** de rama master local a origin
  - \$ `git push -u origin master` # **-u "tracking reference"**

Click aquí: Instrucciones



# Fork: Copiar un proyecto en GITHUB

# Fork permite copiar un repositorio (proyecto) en nuestra cuenta en GITHUB  
#  
# -> Una vez copiado (clonado) tenemos acceso a él y podemos evolucionarlo



The screenshot shows the GitHub interface for the jQuery repository. A red dashed arrow originates from the text in the box above and points to the 'Fork' button in the repository header. The repository page displays various statistics and a list of recent commits.

Repository: jquery / jquery

Watch 2,892 | Star 34,002 | Fork 8,200

jQuery JavaScript Library <http://jquery.com/>

5,784 commits | 8 branches | 125 releases | 216 contributors

branch: master | jquery / +

Docs: Add info about Sizzle not being excludable on the compat branch

mzgol authored 8 days ago | latest commit 062b5267d0

File	Commit Message	Time Ago
build	Deferred: Backwards-compatible standards interoperability	18 days ago
external	Build: Update native-promise-only (again)	10 days ago
src	Attributes: revert returning null for non-existent attributes	8 days ago
test	Attributes: revert returning null for non-existent attributes	8 days ago
.editorconfig	Misc: Need for speed removed by 9ad6e7e	6 months ago
.gitattributes	Build: change .gitattributes; use system line ends for non-JS files	a year ago
.gitignore	Build: drop bower; use npm for front-end deps	9 months ago
.jscsrc	Build: Move all external libraries to external directory	10 months ago
.jshintignore	Build: Move all external libraries to external directory	10 months ago

Code | Issues 101 | Pull requests 12 | Wiki

Pulse | Graphs

HTTPS clone URL: <https://github.com/jquery/jquery.git>

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop | Download ZIP

# Contribuir a un proyecto GITHUB

# La forma habitual de contribuir a un proyecto en GITHUB es seguir estos 4 pasos:

# 1) Crear una copia del repositorio original en GITHUB con "Fork" en la cuenta propia

# 2) Clonar la rama creada en nuestra cuenta en nuestro ordenador local

```
p1> git clone https://github.com/pepe/proy1
```

# 3) Modificar el proyecto local, realizar commit y "push" a nuestra copia en GITHUB

```
p1> .....
```

```
p1> git add ...
```

```
p1> git commit -m '.....'
```

```
p1> git push origin master
```

# 4) Hacer "Pull Request" desde nuestra cuenta en GITHUB pidiendo al administrador del repositorio original que introduzca nuestros cambios

github SOCIAL CODING

jqemada Dashboard Inbox 1 Account Settings Log Out

Explore GitHub Gist Blog Help Search...

lifo / docrails

Watch Fork Pull Request 585 230

Code Network Pull Requests 0 Wiki 5 Stats & Graphs

PLEASE CHECK <http://github.com/lifo/docrails/wikis> — Read more

<http://weblog.rubyonrails.org/2008/5/2/help-improve-rails-documentation-on-git-branch>

Clone in Mac ZIP SSH HTTP Git Read-Only `git@github.com:lifo/docrails.git` Read+Write access

Files Commits Branches 1 Tags Downloads

Current branch: master

Latest commit to the master branch



# GIT

4. Crear proyecto "random" con GITHUB App

# Crear un proyecto con GITHUB App

GITHUB-for-MAC/Windows es una herramienta gráfica muy eficaz para gestionar proyectos git localmente en el PC. Solo está soportada para MAC y para Windows:

Descargar (e instalar) GITHUB for MAC: <https://mac.github.com>

Descargar (e instalar) GITHUB for Windows: <https://windows.github.com>

En este ejemplo vamos a crear desde cero un proyecto, de nombre "random", con 2 versiones de los ejemplos que generan números aleatorios, usados para ilustrar la sentencia if/else. Este proyecto se puede encontrar en GITHUB en: <https://github.com/jquemada/random>

**Desplegable para:**

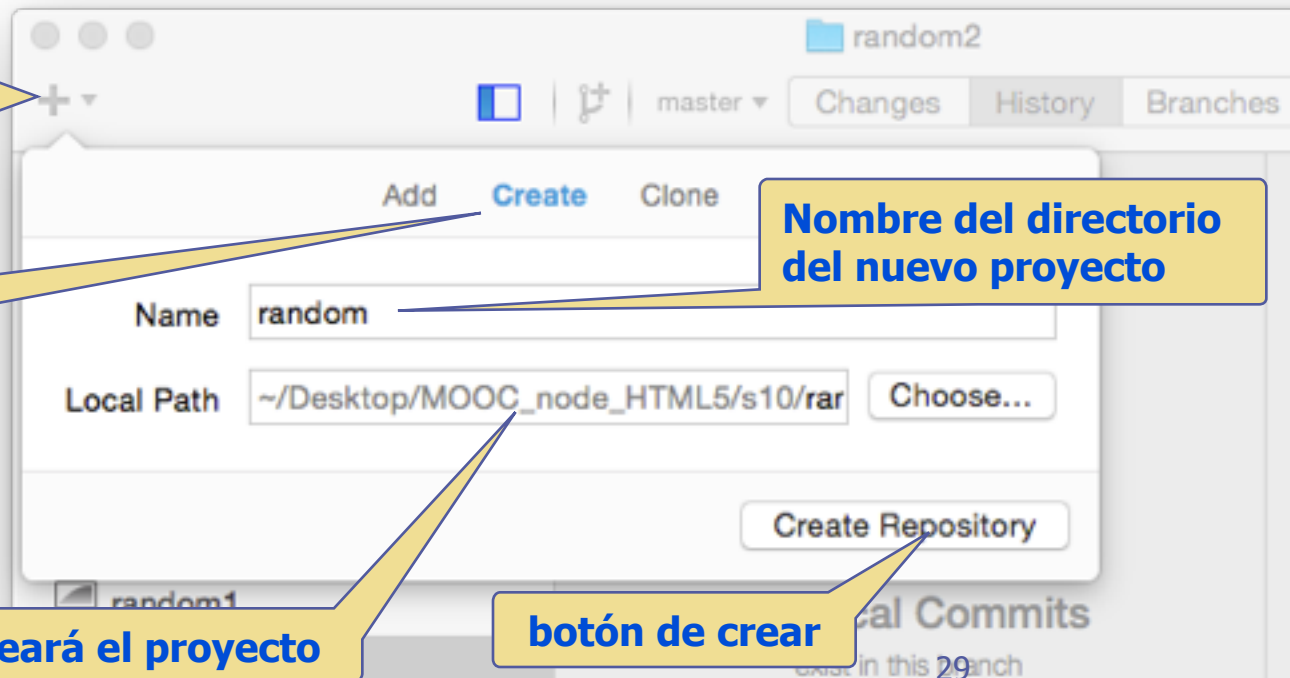
- Añadir (Add) proyecto
- Crear (Create) proyecto
- Clonar (Clone) proyecto

**Pestaña de crear nuevo proyecto**

**Nombre del directorio del nuevo proyecto**

**directorio donde se creará el proyecto**

**botón de crear**



# Crear ficheros del proyecto con Atom

**Nuevo proyecto creado**

Atom es un editor muy sencillo y eficaz para editar los ficheros del proyecto. Una vez instalado Atom, podemos abrirlo y crear nuevos ficheros con "New File".

Click con Botón derecho del ratón (^Click) en proyecto despliega esta ventana de acceso

File	Edit	Selection	Find
New Window		⇧⌘N	
New File		⌘N	
Open...		⌘O	
Reopen Last Item		⇧⌘T	
Save		⌘S	
Save As...		⇧⌘S	
Save All		⌘⌘S	
Close Tab		⌘W	
Close Pane			
Close Window		⇧⌘W	

# Crear/Editar fichero random.js

The image shows two screenshots from a macOS environment. The top screenshot shows the Atom text editor with a file named `random.js` open. The code inside is as follows:

```
1 // Math.random() devuelve número aleatorio entre 0 y 1.
2 var numero = Math.random();
3
4 if (numero <= 0.5){
5   console.log('\n' + numero + ' MENOR que 0,5 \n');
6 }
7 else {
8   console.log('\n' + numero + ' MAYOR que 0,5 \n');
9 }
10
```

The bottom screenshot shows the GitHub Desktop application. On the left, the 'Uncommitted Changes' panel shows a new file `random.js` ready to be committed. A yellow callout box points to this file with the text: "Primer ejemplo de número aleatorio con Math.random() e if/else, que se guarda en la versión 1 del proy random." In the center, the 'Commit to master' dialog is open, and a yellow callout box points to the 'Commit' button with the text: "Area de trabajo del proyecto con los cambios realizados." On the right, the code for `random.js` is displayed, matching the code in the Atom editor. A red arrow points from the 'Save As...' option in the Atom 'File' menu to the `random.js` file in the GitHub Desktop interface.

**File** Edit Selection Find

- New Window ⌘N
- New File ⌘N
- Open... ⌘O
- Reopen Last Item ⌘T
- Save ⌘S
- Save As... ⌘S
- Save All ⇧⌘S
- Close Tab ⌘W
- Close Pane
- Close Window ⇧⌘W

random.js 10,1 UTF-8 JavaScript +10

random

Changes History Branches Settings

Committing to master

Uncommitted Changes

Commit

Select All

NEW .gitignore

NEW random.js

~/Desktop/MOOC\_node\_HTML5/s10/random

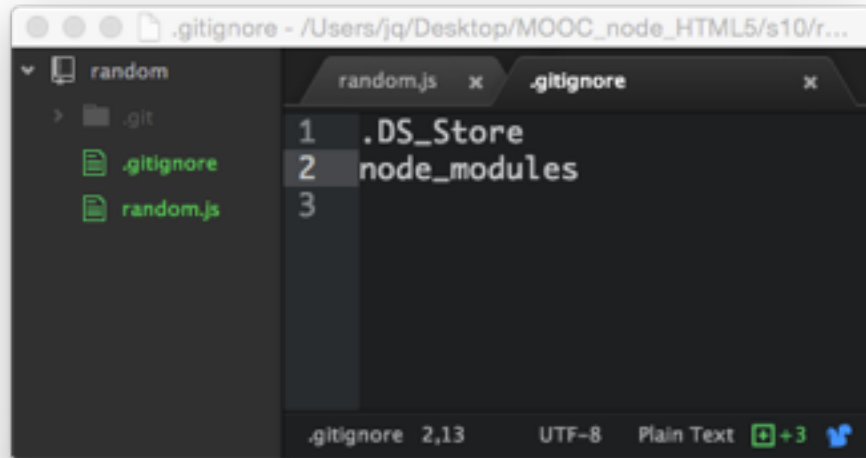
random.js

```
... @@ -0,0 +1,9 @@
1 + // Math.random() devuelve número aleatorio entre 0 y 1.
2 + var numero = Math.random();
3 +
4 + if (numero <= 0.5){
5 +   console.log('\n' + numero + ' MENOR que 0,5 \n');
6 + }
7 + else {
8 +   console.log('\n' + numero + ' MAYOR que 0,5 \n');
9 + }
```

Primer ejemplo de número aleatorio con Math.random() e if/else, que se guarda en la versión 1 del proy random.

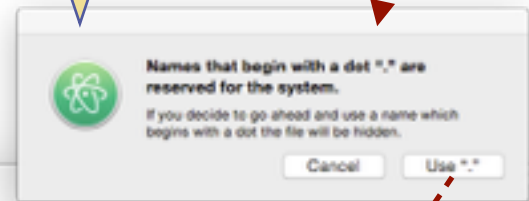
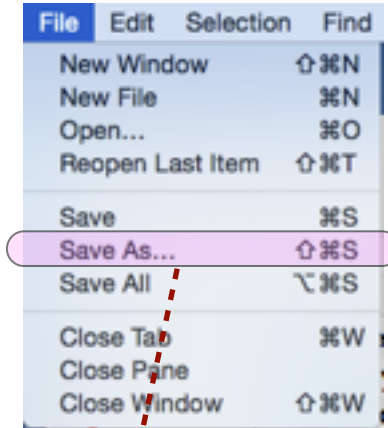
Area de trabajo del proyecto con los cambios realizados.

# Crear/Editar fichero .gitignore



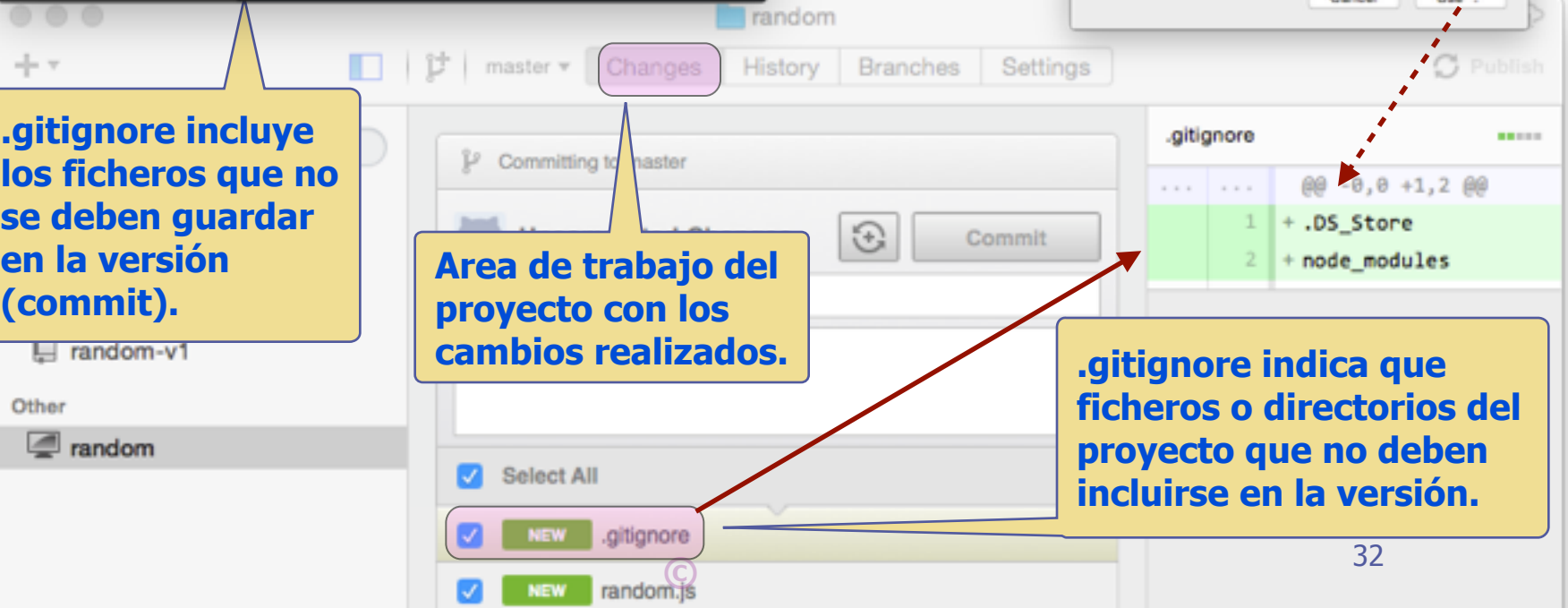
**.gitignore incluye los ficheros que no se deben guardar en la versión (commit).**

**Como .gitignore es un fichero del sistema, al guardarlo nos pide confirmación.**



**Area de trabajo del proyecto con los cambios realizados.**

**.gitignore indica que ficheros o directorios del proyecto que no deben incluirse en la versión.**





# Crear version 1 (commit 1)

La **versión (commit)** se genera en el **proyecto local** guardado en el **directorio de nuestro ordenador** que se creó cuando se creó el proyecto.

El área de cambios se vacía al generar versión. Los cambios de la nueva versión aparecen al modificar ficheros.

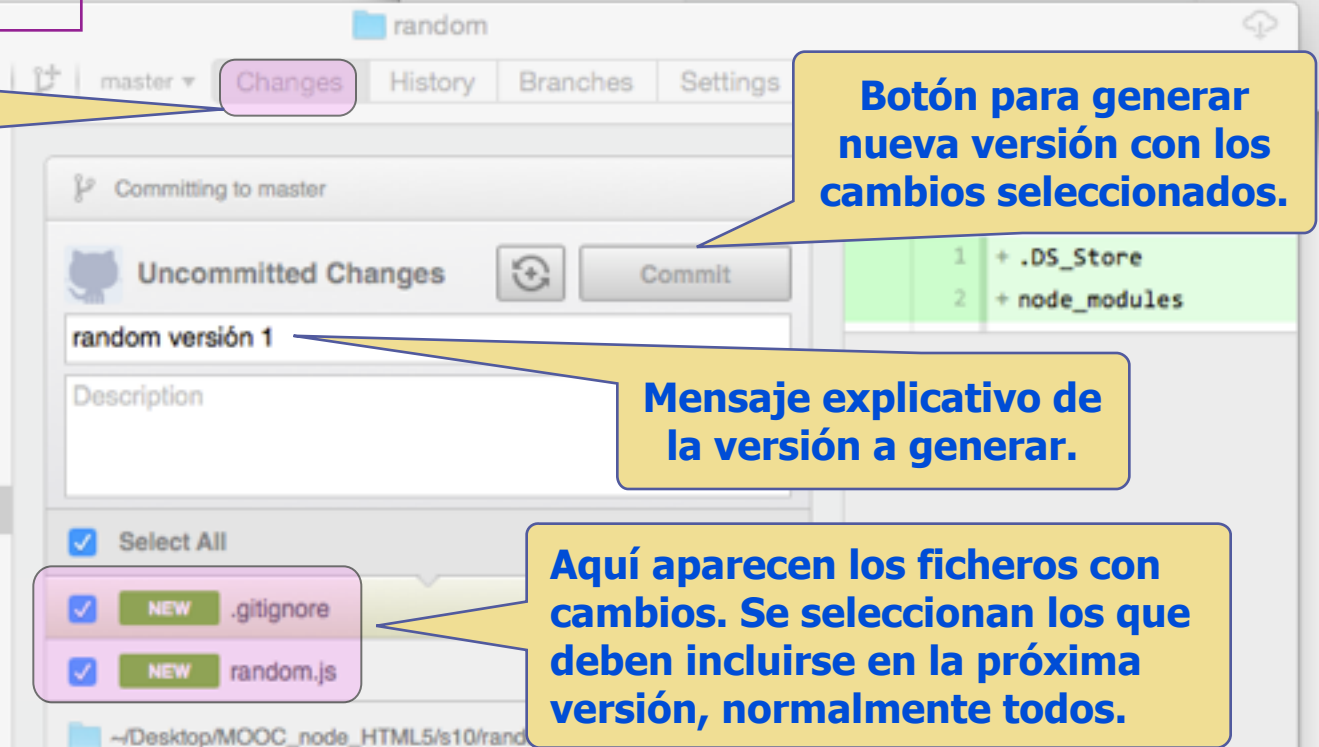
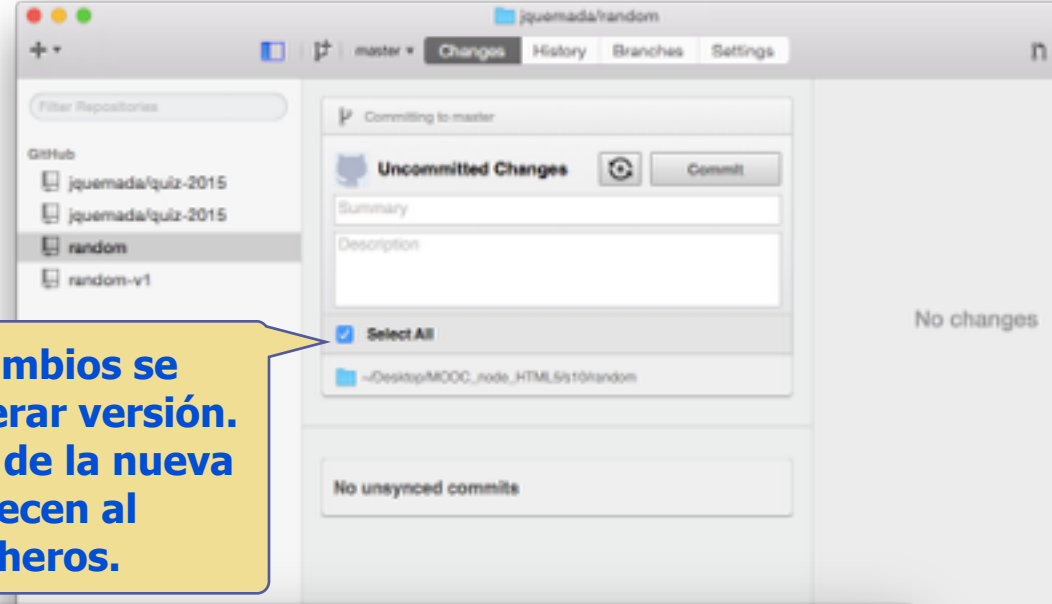
Área de trabajo del proyecto con los cambios realizados.

Botón para generar nueva versión con los cambios seleccionados.

Mensaje explicativo de la versión a generar.

Aquí aparecen los ficheros con cambios. Se seleccionan los que deben incluirse en la próxima versión, normalmente todos.

Proyecto



# Definir repositorio remoto "origin"

Una **cuenta en GITHUB** con credenciales de acceso nos permite asignar al **repositorio remoto primario "origin"** un repositorio creado **GITHUB**. Para ello asignamos el **URL de un repositorio vacío creado en GITHUB** al repositorio primario remoto **"origin"**.

Los URLs de repositorios en otras cuentas serán diferentes:  
<https://github.com/jquemada/random> pertenece a "jqumada".

Configurar repositorio origin y .gitignore (se configura en pestaña "Repository" también).

Botón para configurar "origin"

Proyecto random

Primary remote repository (origin)

<https://github.com/jquemada/random>

Update Remote

"origin" es un repositorio remoto en GITHUB que identificamos por un URL (<https://github.com/jquemada/random>) donde guardar/publicar el proyecto.

Ignored files (.gitignore)

.DS\_Store  
node\_modules

.gitignore se puede configurar aquí también. Es una forma alternativa a la ya mostrada.

# Publicar rama "master" en "origin"

Un proyecto se realiza siempre en una **rama de desarrollo**, donde se **guardan las versiones que se generan**. La **rama "master"** se crea por defecto al crear un proyecto y existe siempre. Este ejemplo **guarda todo en la rama master**. Mas adelante se ve como gestionar otras ramas.

The screenshot shows the GitHub Desktop application window for a repository named 'jqemada/random'. The 'Branches' tab is selected in the top navigation bar. The left sidebar shows a list of repositories and branches, with 'random' selected. The main area displays the 'master' branch details, including the commit hash '31727a2', the commit message 'Juan Quemada: random versión 1', the time '6:17 am', and a 'Published' button. Three callout boxes are present: one pointing to the 'Branches' tab, one pointing to the branch list, and one pointing to the 'Published' button.

**Area de gestión de ramas**

En este área se ve el estado de las ramas del proyecto. En este ejemplo solo tenemos la rama "master" que se creo por defecto al crear el proyecto.

Botón para publicar "origin" en GITHUB

# Historia de la rama master del proyecto

La historia de cada rama de un proyecto es la **secuencia de versiones generadas**. Seleccionando una versión se visualizan los **cambios realizados en una versión respecto a la anterior**: el **código añadido se muestra en verde** y el **código eliminado se muestra en rojo**.

Rama master seleccionada

Area de gestión de historia del proyecto

Solo se ha generado la primera versión, identificada por el mensaje asociado.

Cambios de esta versión respecto a la anterior. Al ser la primera versión el código de los 2 ficheros creados es verde.

The screenshot displays the GitHub web interface for the 'jqemada/random' repository. The 'History' tab is active, showing a single commit titled 'random versión 1' by user 'jqemada' 7 minutes ago. The commit message is 'random versión 1'. The diff view on the right shows two new files: '.gitignore' and 'random.js'. Both files are highlighted in green, indicating they were added in this commit. The '.gitignore' file contains entries for '.DS\_Store' and 'node\_modules'. The 'random.js' file contains a JavaScript function that generates a random number between 0 and 1 and logs it to the console.

```
random versión 1
jqemada 31727a2 7 minutes ago

.gitignore
@@ -0,0 +1,2 @@
+ .DS_Store
+ node_modules

random.js
@@ -0,0 +1,9 @@
+ // Math.random() devuelve número aleatorio entre 0 y 1.
+ var numero = Math.random();
+
+ if (numero <= 0.5){
+   console.log('\n' + numero + ' MENOR que 0,5 \n');
+ }
+ else {
+   console.log('\n' + numero + ' MAYOR que 0,5 \n');
+ }
```

# Crear nueva versión

Segundo ejemplo de número aleatorio con `Math.random()` e if/else.

1. Abrir proyecto en Atom, seleccionar ficheros y hacer los cambios.

Area de trabajo

3. Generar versión

2. Seleccionar cambios e introducir mensaje

Código añadido en verde, código eliminado en rojo y resto en blanco.

File	Edit	Selection	Find
New Window			⇧⌘N
New File			⇧⌘N
Open...			⇧⌘O
Reopen Last Item			⇧⌘T
Save			⌘S
Save As...			⇧⌘S
Save All			⇧⌘S
Close Tab			⌘W
Close Pane			
Close Window			⇧⌘W

random.js - /Users/jq/Desktop/MOOC\_node\_HTML5/s10/random - Atom

```
1 // Math.random() devuelve número aleatorio entre 0 y 1.
2 var numero = Math.random();
3
4 var str = ' MAYOR que 0,5';
5
6 if (numero <= 0.5){
7   str = ' MENOR que 0,5';
8 }
9
10 console.log('\n' + numero + str + '\n');
11
```

random.js 10,41

UTF-8

JavaScript

master

+5, -4

Changes

Uncommitted Changes

Commit

random versión 2

Description

☒ Select All

☒ random.js

View on GitHub  
Open in Finder  
Open in Terminal  
Open in Atom

Remove

No unsynced commits

# Historia de la rama master del proyecto

La historia de la rama master muestra ahora las 2 versiones generadas. Como se ha seleccionado la última, ahora se muestran los cambios respecto a la primera: el **código añadido se muestra en verde** y el **código eliminado se muestra en rojo**.

The screenshot shows a Git client interface with the following elements:

- Top Bar:** Includes a dropdown menu showing 'master', tabs for 'Changes', 'History', 'Branches', and 'Settings', and a 'Sync' button.
- Left Panel:** A sidebar with a 'Filter Repositories' search bar and a list of repositories: 'jquemada/quiz-2015', 'jquemada/quiz-2015', 'random' (selected), and 'random-v1'.
- Commit History:** A list of commits under the heading '2 commits'. The first commit is 'random versión 2' by 'jqemada' (Just now), which is selected with a blue bar and a radio button. The second commit is 'random versión 1' by 'jqemada' (1 hour ago).
- Diff View:** The right pane shows the diff for 'random versión 2'. It displays the file 'random.js' with line-by-line changes. Lines 4, 5, 7, 8, 9, and 10 are highlighted in green, indicating additions. Lines 5, 6, 7, and 8 are highlighted in red, indicating deletions. The code shows a function that generates a random number and logs it with a message.

Annotations (callouts) are present:

- Rama master seleccionada:** Points to the 'master' branch in the top bar.
- historia del proyecto:** Points to the 'History' tab.
- Sincronizar "origin":** Points to the 'Sync' button.
- Se selecciona la versión 2 para mostrar cambios con versión anterior:** Points to the selected commit 'random versión 2'.
- Aparecen 2 versiones en la historia:** Points to the list of commits.

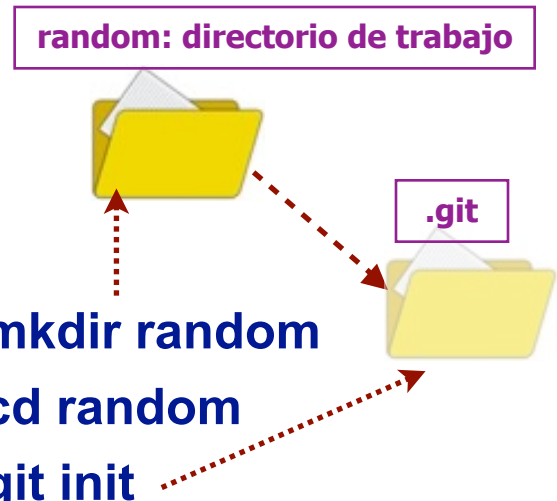


# GIT

5. Crear proyecto "random" con comandos

# Crear proyecto por comando

- ◆ Paso 1. Crear el directorio del proyecto -> **mkdir random**
- ◆ Paso 2. Entrar en directorio del proyecto -> **cd random**
- ◆ Paso 3. Inicializar repositorio git en el directorio -> **git init**
- ◆ “**git init**” habilita un directorio como repositorio de un proyecto
  - Los comandos git deberán ejecutarse en el directorio
- ◆ “**git init**” crea el subdirectorio oculto **.git** con el repositorio
  - El repositorio contendrá las versiones guardadas en un proyecto



```
random — bash — 62x6
venus:~ jq$
venus:~ jq$ mkdir random
venus:~ jq$ cd random
venus:random jq$ git init
Initialized empty Git repository in /Users/jq/random/.git/
venus:random jq$
```



# El directorio de trabajo

## ◆ Directorio de trabajo (working directory)

- Contiene todos los ficheros del proyecto
  - El contenido del directorio cambia a medida que el proyecto avanza

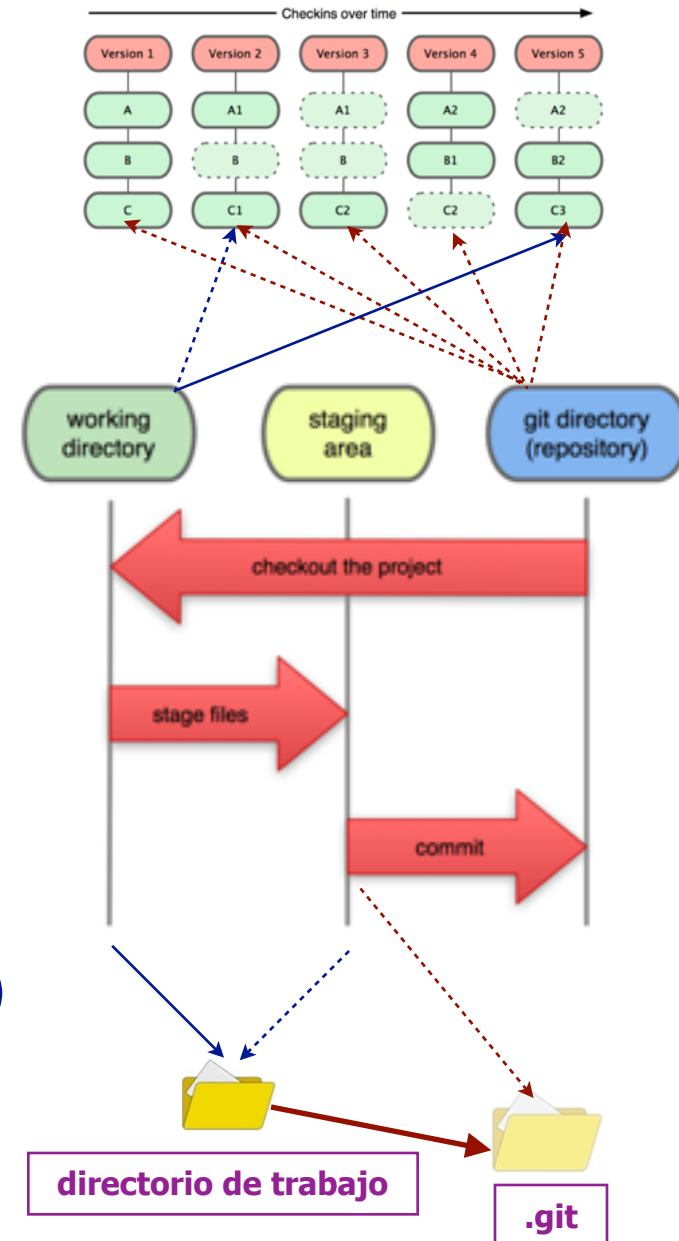
## ◆ Área de cambios o índice (staging area, index)

- Ficheros indexados para la próxima versión
  - Serán **ficheros borrados, nuevos o modificados** respecto a la **versión anterior**
    - "git add ...." añade al índice
    - "git commit ...." crea versión
  - **OJO!** Un fichero **modificado pero no indexado** no se incluirá en la versión

## ◆ Repositorio GIT (Directorio oculto ".git")

- Contiene todas las versiones del proyecto
  - "git checkout ...."
    - reconstruye (descongela) versiones del proyecto en el directorio de trabajo (working directory)

\*Scott Chanson: <http://git-scm.com/book>



# .gitignore

# **.gitignore** es un fichero que informa de los ficheros que no debe gestionar GIT.  
# - **git status** no los presentará como ficheros untracked.  
# - **git add .** no los añadirá al staging area.

# Los ficheros **.gitignore** pueden crearse en cualquier directorio del proyecto,  
# y afectan a ese directorio y a sus subdirectorios.

# Su contenido: líneas con patrones de nombres.

# - Puede usarse los comodines **\*** y **?**

# - Patrones terminados en **/** indican directorios

# - Un patron que empiece con **!** indica negación

# - Se ignoran líneas en blanco y que comiencen con **#**

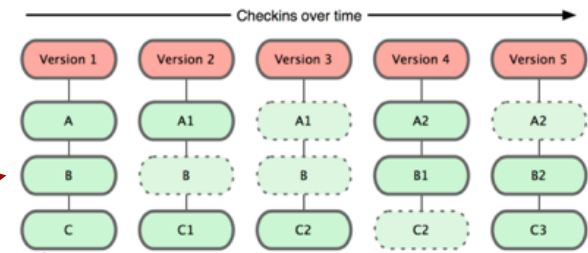
# - **[abc]** indica cualquiera de los caracteres entre corchetes

# - **[a-z]** indica cualquiera de los caracteres en el rango especificado

# Ejemplo

<b>private.txt</b>	# excluir los ficheros con nombre "private.txt"
<b>*.class</b>	# excluir los ficheros acabados en ".class"
<b>*.[oa]</b>	# excluir ficheros acabados en ".o" y ".a"
<b>!lib.a</b>	# no excluir el fichero "lib.a"
<b>*~</b>	# excluir ficheros acabados en "~"
<b>testing/</b>	# excluir directorio "testing"

# Creación de un proyecto y sus versiones



```
$ git init # Se inicia proyecto, creando repositorio vacío en .git/
$ .....
$ git add random.js # añade fichero random.js al índice
$ .....
$ git add . # Crear .gitignore con ficheros no indexados
$ git commit -m 'random versión 1' # añade resto de cambios a índice
# congela 1a versión

$ ..... # se modifica random.js
$ git add . # añade cambios a índice
$ git commit -m 'random versión 2' # congela 2a versión

# Creamos una cuenta en GITHUB y un repositorio vacío random para subir el proyecto

# Asociamos "origin" a repositorio remoto en GITHUB https://github.com/jquemada/random
$ git remote add origin https://github.com/jquemada/random
$ git push origin master # subimos la rama "master" a repositorio remoto "origin"
.....
# Clonamos repositorio remoto https://github.com/jquemada/random en directorio random-2
$ git clone https://github.com/jquemada/random random-2

$ cp -r random random-3 # random puede copiarse. random-3 sera otro repo. independiente
```

# Crear nuevas versiones: add y commit

## ◆ git add .....

- añade fichero(s) al índice para próxima versión
  - **git add .** -> añade todo lo modificado al índice
  - **git add file\_1.js file\_2.js** -> añade solo ficheros file1.js y file2.js
    - **Ojo:** cambios posteriores a invocar “**git add ..**” no se añaden al índice

## ◆ git commit -m “mensaje”

- crea nueva versión en la rama actual, incluye lo registrado en el índice
  - **-m “mensaje”** incluye un mensaje que identifica la versión

### # Ayuda en línea de comandos

**\$ git init --help**      # muestra ayuda en línea (manual) de “git init”

.....

**\$ git add --help**      # muestra ayuda en línea (manual) de “git add”

.....

**\$ git commit --help**    # muestra ayuda en línea (manual) de “git commit”

# Modificar el último "commit"

```
# Para modificar el último commit usaremos git commit --amend -m ...  
# Para cambiar el mensaje de log.  
# Para añadir una modificación olvidada  
# ...
```

```
$ git commit -m 'editor acabado' # creamos el commit pero olvidamos  
# añadir un fichero, y el mensaje de  
# log no esta en ingles
```

```
$ ..... # Realizamos los cambios olvidados  
$ git add forgotten_file # y los subimos al índice
```

```
# Repetimos git commit con opción --amend y un mensaje de log (modificado o no)  
$ git commit --amend -m "editor acabado"
```

```
# Se actualiza el commit erróneo con los nuevos cambios introducidos
```

**IMPORTANTE:** no realizar --amend sobre un commit que se haya hecho público a otros desarrolladores (publicado en otro repositorio).

# git log: Historia de versiones

```
# La historia de versiones (commits) de de la rama en la que se está trabajando  
# -> se muestra con "git log"  
# "git log --stat"           # muestra estadísticas  
# "git log --graph"          # muestra árbol  
# "git log --since=2.weeks"  # muestra commits últimas 2 semanas  
# "git log --oneline"        # muestra resumen de cada commits en 1 línea  
# "git log -5"              # muestra 5 últimos commits
```

```
$ git log -2      # Muestra 2 últimos commits  
commit b48cd0b84dd71d4314b11a917f2971b26b464d92  
Author: Juan Quemada <jquemada@dit.upm.es>  
Date: Thu Apr 2 13:13:15 2015 +0200
```

random versión 2

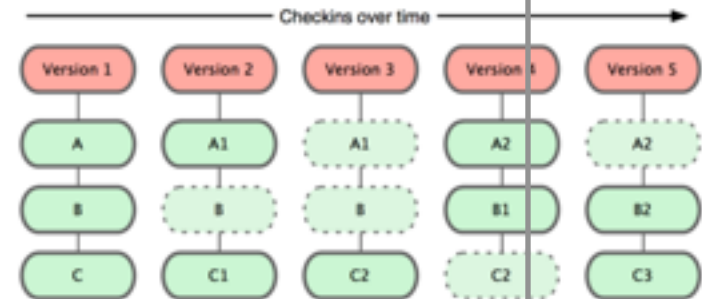
```
commit b66f1fb6c70f3f669b216fc25aac0f5ebe1542f2  
Author: Juan Quemada <jquemada@dit.upm.es>  
Date: Thu Apr 2 13:08:47 2015 +0200
```

random versión 1

```
$ git log --oneline      # Muestra resumen de 1 línea de commits
```

```
b48cd0b random versión 2  
b66f1fb random versión 1
```

```
$
```



# La historia de diferencias entre commits se muestra con opción -p, por ejemplo "git log -p -1"

```
$ git log -p -1
commit 188799e21c4a71f13a2e729ccf77f3a960885682
Author: Juan Quemada <jquemada@dit.upm.es>
Date: Mon Nov 21 18:17:13 2011 +0100
```

migración base de datos

## Diferencias entre versiones

```
diff --git a/db/schema.rb b/db/schema.rb
index b5e6a79..61dcaab 100644
```

```
--- a/db/schema.rb
```

```
+++ b/db/schema.rb
```

```
@@ -11,6 +11,13 @@
```

```
#
```

```
# It's strongly recommended to check this file into your version control system.
```

```
-ActiveRecord::Schema.define(:version => 0) do
```

```
+ActiveRecord::Schema.define(:version => 20111121171513) do
```

```
+
```

```
+ create_table "types", :force => true do |t|
```

```
+   t.string "name"
```

```
+   t.text "description"
```

```
+   t.datetime "created_at"
```

```
+   t.datetime "updated_at"
```

```
+ end
```

```
end
```

```
$
```

# Rama master y puntero HEAD

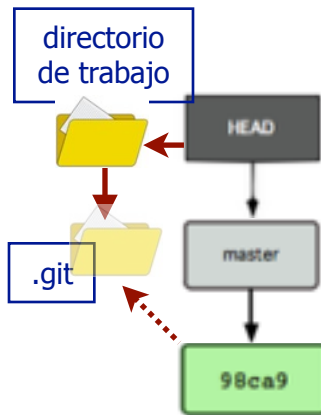
♦ **master** es la rama principal del desarrollo

- “**git init**” inicia el proyecto en la **rama master**
  - Las versiones (commits) se crean en **master** (salvo que se pase a otra rama)
- **master** es un puntero a la última versión de esta rama principal

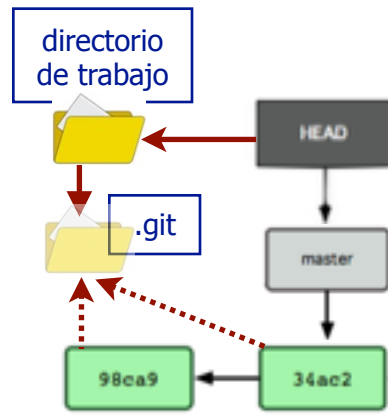
♦ HEAD referencia la versión (commit) actual del directorio de trabajo

♦ Cada “**git commit ...**” crea una nueva versión

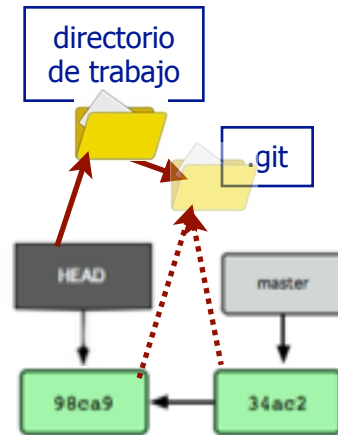
- actualizando los **punteros master y HEAD**



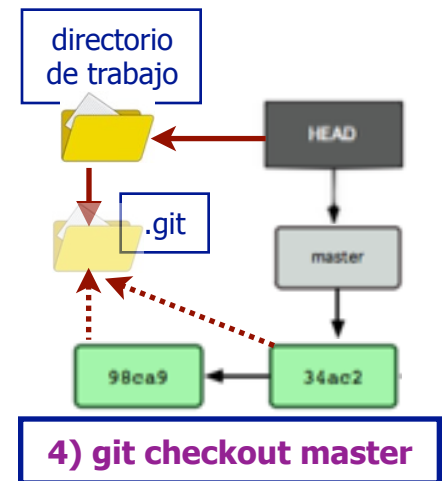
1) **git commit -m “..1”**



2) **git commit -m “..2”**



3) **git checkout 98ca9**



4) **git checkout master**

**OJO! Detached head: peligroso**



# Reset: Eliminar commits

# "git reset <commit\_id>"

# -> vuelve a <commit\_id> eliminando versiones posteriores

\$ git log --oneline # lista commits

c2b9e migración base de datos

f30ab creación de scaffold Type

34ac2 añadir ejemplo

98ca9 vistas index y contact



\$ git reset 34ac2

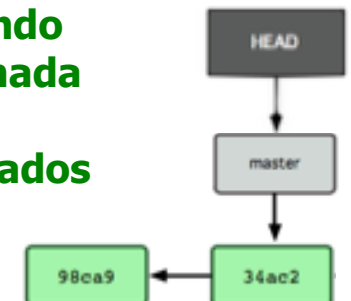
# restaura versión 34ac2 'añadir ejemplo' dejando los cambios  
# realizados en las versiones eliminadas en directorio de trabajo  
# sin añadir al índice (staging-area)

\$ git reset --hard 34ac2

# restaura 34ac2 'añadir ejemplo' eliminando  
# todos los cambios de las versiones eliminada

# ¡OJO! con "commit reset --hard ..." se pierden los commits eliminados

\$





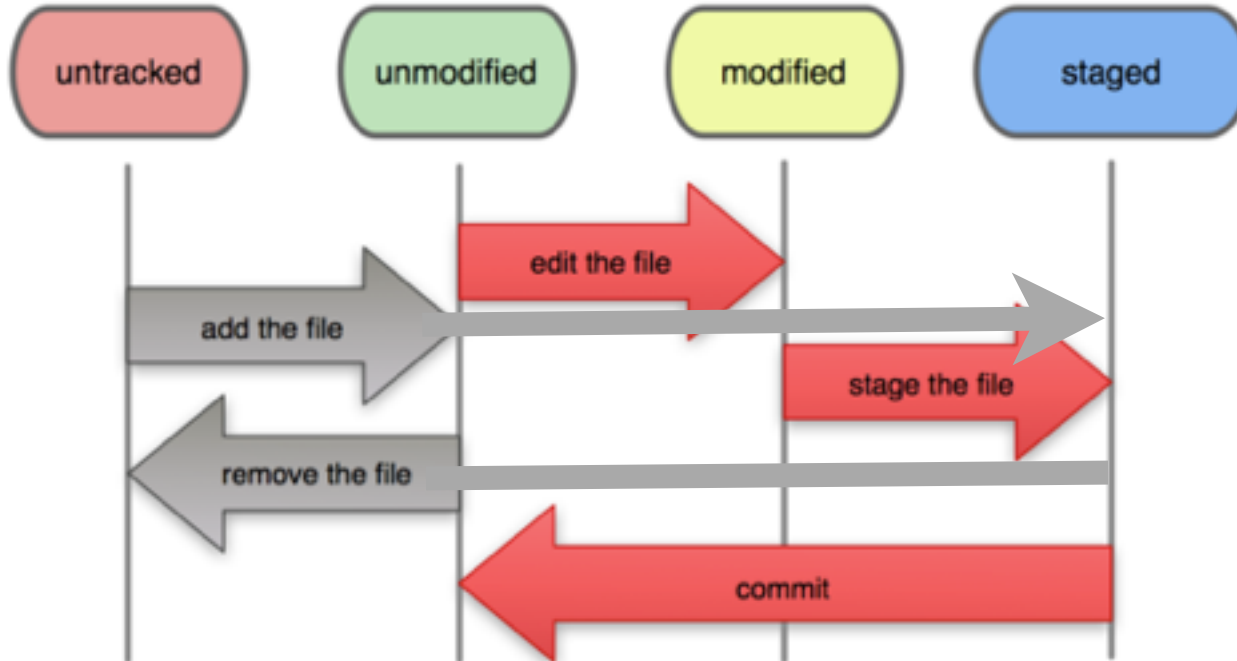
# GIT

## 6. Análisis y gestión del área de trabajo

# Estado de los ficheros

◆ Los ficheros del directorio de trabajo pueden estar

- **Untracked:** Ficheros que no están bajo el control de versiones
- **Tracked:** Ficheros registrados en versión (con **git add .....**)
  - **Modified:** ficheros modificados, no incluidos en próximo commit con **git add ...**
  - **Unmodified:** ficheros no modificados, que **siguen en próximo commit**
  - **Staged:** ficheros modificados, incluidos en próximo commit con **git add...**
- **Ignorados:** Ficheros indicados en **.gitignore**



The diagram illustrates the lifecycle of a file in a version control system like Git. It features four colored ovals at the top representing different states: **untracked** (red), **unmodified** (green), **modified** (yellow), and **staged** (blue). Below these, a series of arrows show the transitions between states:

- A grey arrow labeled "add the file" points from the **untracked** state to the **unmodified** state.
- A grey arrow labeled "remove the file" points from the **unmodified** state back to the **untracked** state.
- A red arrow labeled "edit the file" points from the **unmodified** state to the **modified** state.
- A red arrow labeled "stage the file" points from the **modified** state to the **staged** state.
- A red arrow labeled "commit" points from the **staged** state back to the **unmodified** state.

Two vertical dashed red lines are positioned between the **untracked** and **unmodified** states, and between the **modified** and **staged** states.

```
# 1) Changes to be committed: ficheros modificados indexados con "git add ..."
# 2) Changed but not updated: ficheros modificados no indexados con "git add ..."
# 3) Untracked files: ficheros nuevos no indexados con "git add ..." o extraídos con "git rm ..."
```

## Ficheros modificados incluidos en próxima versión

## Ficheros modificados no incluidos en próxima versión

## Ficheros excluidos de versión

# Borrar ficheros

# Eliminar un fichero en la próxima versión a congelar:

\$ **git rm** CharIO.java    # Borra el fichero del directorio de trabajo y del staging area.  
# Tras el próximo commit dejará de estar tracked.

\$ **git rm --cached** CharIO.java    # Borra fichero del staging area.  
# No lo borra del directorio de trabajo.  
# Tras el próximo commit dejará de estar tracked.

# El comando del S.O. **rm** borra ficheros del directorio de trabajo,  
# pero no los borra del staging area.  
# Es como hacer una modificación en el contenido del fichero.  
# Debe usarse **git add** o **git rm** para meter en el staging area esta modificación.

\$ **rm** CharIO.java    # borra el fichero de directorio de trabajo,  
# pero este cambio aun no ha sido staged.

# **git rm** falla si se intenta borrar un fichero con modificaciones en el directorio  
# de trabajo o en el staging area (índice).  
# Para no perder de forma accidental modificaciones realizadas.  
# Usar la opción **-f** para forzar el borrado.

# Renombrar ficheros

**# Mover o renombrar un fichero:**

```
$ git mv filename_old filename_new
```

```
$ git mv index.htm index.html
```

**# Internamente se implementa ejecutando los comandos git rm y git add**

```
$ git mv filename_old filename_new
```

**# es equivalente a ejecutar:**

```
$ mv filename_old filename_new
```

```
$ git rm filename_old
```

```
$ git add filename_new
```

The diagram illustrates the lifecycle of a file in Git across four states: untracked (red), unmodified (green), modified (yellow), and staged (blue). The transitions are as follows:

- add the file:** Moves a file from the **untracked** state to the **unmodified** state.
- edit the file:** Moves a file from the **unmodified** state to the **modified** state.
- stage the file:** Moves a file from the **modified** state to the **staged** state.
- remove the file:** Moves a file from the **unmodified** state back to the **untracked** state.
- commit:** Moves a file from the **staged** state back to the **unmodified** state.

## \$ git diff

--- a/benchmarks.rb

+++ [b/benchmarks.rb](#)

```
@@ -36,6 +36,10 @@ def main
```

## # rango de líneas con cambios

```
@commit.parents[0].parents[0].parents[0]
```

**end**

## # contenido no modificado, enmarca cambios

- # -> insert new task here

## # líneas eliminadas empiezan por "-"

```
+ run_code(x, 'commits 1') do
```

## # líneas nuevas empiezan por “+”

+ **git.commits.size**

+ end

+

```
run_code(x, 'commits 2') do
```

## # contenido no modificado, enmarca cambios

```
log = git.commits('master', 15)
```

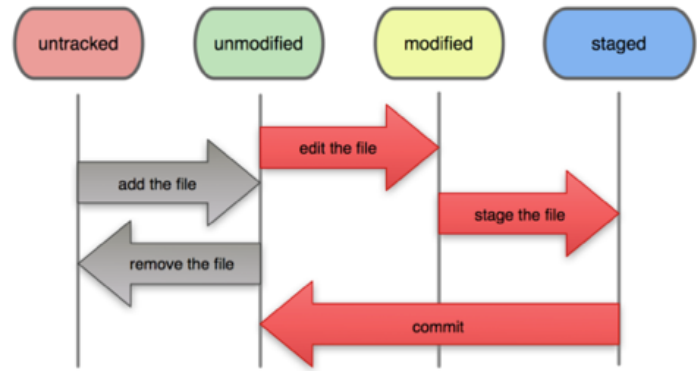
## log.size

```
diff --git a/tests.rb b/tests.rb
```

## # diferencias de fichero tests.rb

████████████████████

# 1. Análisis del estado del área de trabajo: "git diff --cached"



## # "git diff --cached" o git diff --staged" muestra diferencias en ficheros modificados e indexados (staged)

## \$ git diff --staged

```
diff --git a/benchmarks.rb b/benchmarks.rb
```

index 3cb747f..da65585 100644

--- a/benchmarks.rb

+++ [b/benchmarks.rb](#)

```
@@ -36,6 +36,10 @@ def main
```

## # rango de líneas con cambios

```
@commit.parents[0].parents[0].parents[0]
```

end

## # contenido no modificado, enmarca cambios

- **# -> insert new task here**

## # líneas eliminadas empiezan por "-"

```
+ run_code(x, 'commits 1') do
```

## # líneas nuevas empiezan por “+”

+ **git.commits.size**

+ end

+

```
run_code(x, 'commits 2') do
```

## # contenido no modificado, enmarca cambios

```
log = git.commits('master', 15)
```

**log.size**



# Eliminar Modificaciones en el Directorio de trabajo

# Para eliminar las modificaciones realizadas en un fichero del  
# directorio de trabajo, y dejarlo igual que la version del repositorio:  
# **git checkout -- <file>**

# Ejemplo:

# Modificamos un fichero.

\$ **vi readme.txt** # editamos el contenido del fichero readme.txt

# Nos arrepentimos de los cambios realizados.

# Para restaurar el fichero a su estado original ejecutamos:

\$ **git checkout -- readme.txt**

# **"git checkout ."** deshace todos los cambios staged de area de trabajo

\$ **git checkout .**

# Deshacer operaciones realizadas: Eliminar Modificaciones Staged

```
# Para eliminar del staging area las modificaciones de un fichero:  
#  git reset HEAD <file>
```

```
# Ejemplo:
```

```
# Modificamos un fichero y registramos los cambios en el staging area:
```

```
$ vi readme.txt # editamos el contenido del fichero readme.txt  
$ git add .      # añadimos todos los cambios existentes en todos los  
                  # ficheros tracked al staging area.
```

```
# Pero no queremos añadir los cambios de readme.txt.
```

```
# Para cambiar el estado de readme.txt a unstaged ejecutamos:
```

```
$ git reset HEAD readme.txt
```

```
# readme.txt ya no esta modificado en el staging area.
```

```
# readme.txt conserva sus modificaciones en el directorio de trabajo.
```

# GITHUB for MAC & for Windows

GITHUB-for-MAC/Windows permite ver todos los cambios realizados a la última versión en el directorio de trabajo. Además, el editor Atom nos permite hacer cambios con mucha facilidad.

**Directorio de trabajo**

**Proyecto  
seleccionado**

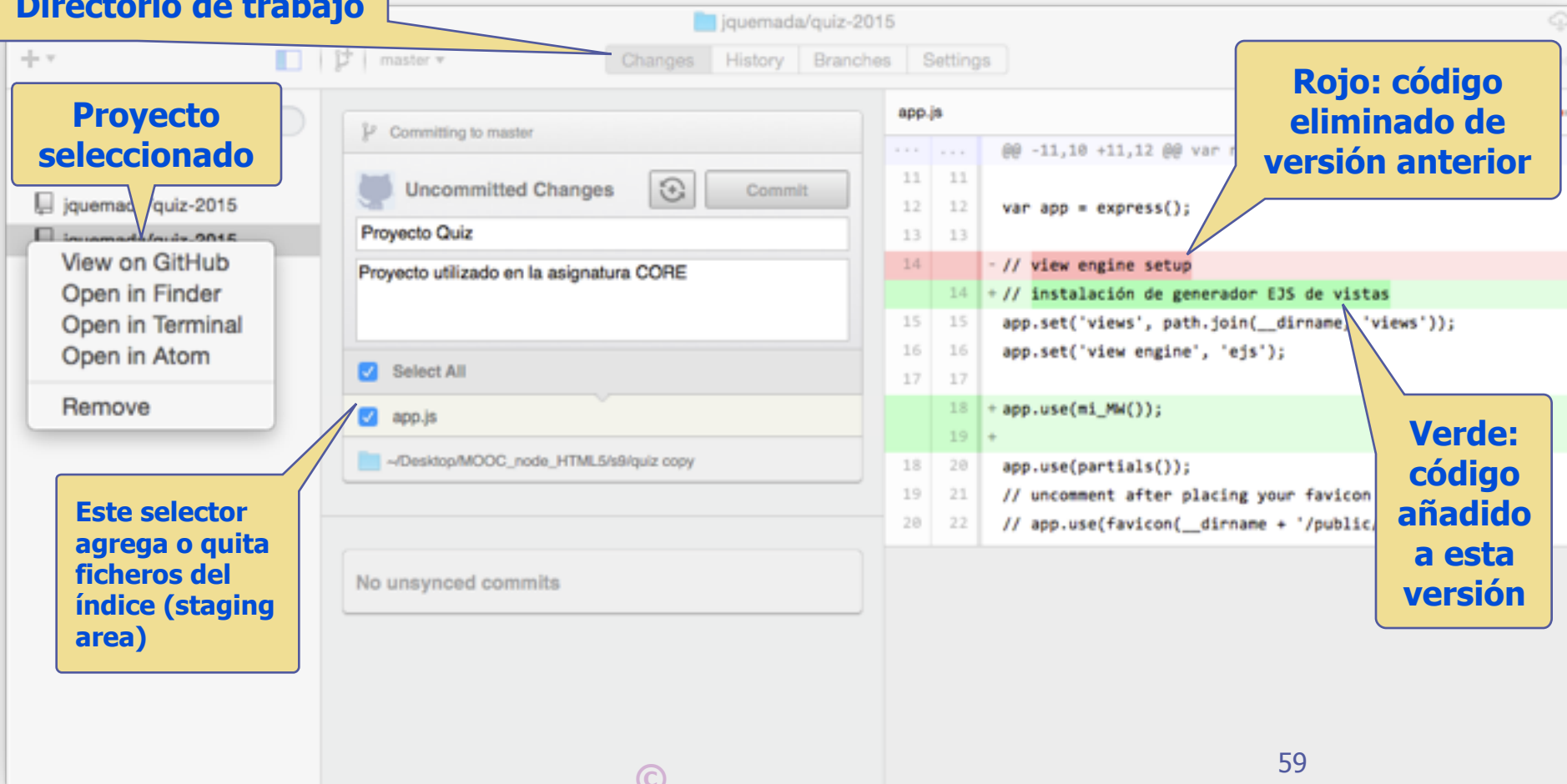
View on GitHub  
Open in Finder  
Open in Terminal  
Open in Atom

Remove

**Este selector  
agrega o quita  
ficheros del  
índice (staging  
area)**

**Rojo: código  
eliminado de  
versión anterior**

**Verde:  
código  
añadido  
a esta  
versión**





# GIT

## 7. Ramas

# Crear ramas

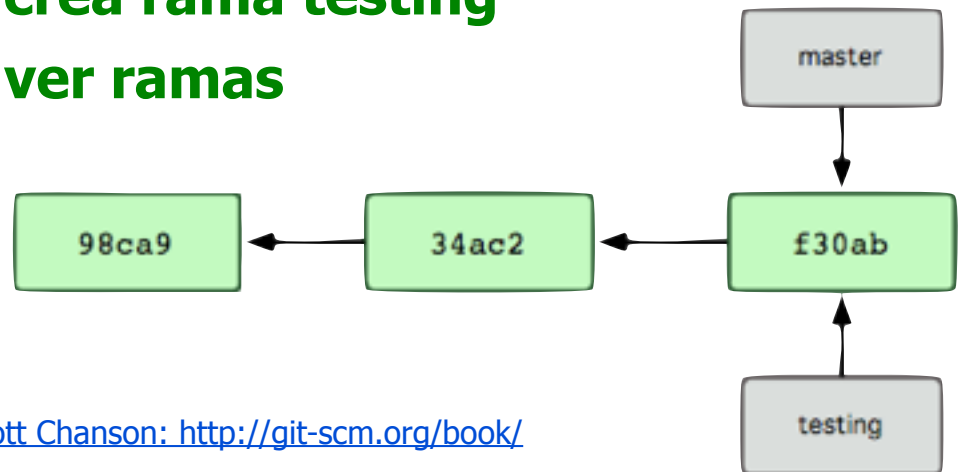
- ♦ **Rama**: desarrollo que diverge de la rama master o de otra rama
  - ★ A partir de alguna versión (commit)
- ♦ Una rama se crea con: **git branch <nombre\_de\_rama>**
  - ★ Crea un nuevo puntero asociado a la rama
- ♦ **git branch** permite ver las ramas creadas
  - ★ \* indica cual esta seleccionada

**\$ git branch testing    # crea rama testing**

**\$ git branch            # ver ramas**

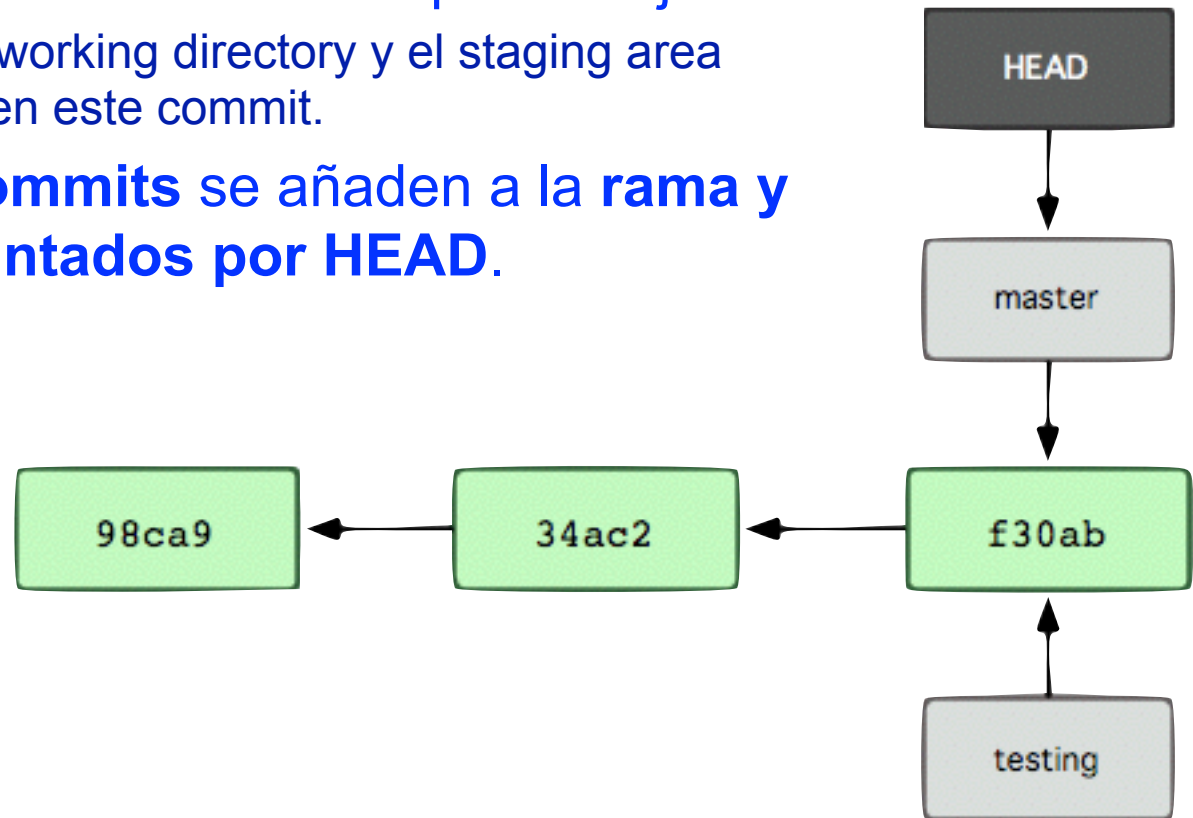
**\* testing  
master**

**\$**



*\*de Scott Chanson: <http://git-scm.org/book/>*

- ◆ En GIT existe una referencia llamada **HEAD** que apunta a la rama actual
  - ★ que apunta al commit sobre el que trabajamos.
    - los ficheros del working directory y el staging area están basados en este commit.
  - ★ Los **nuevos commits** se añaden a la rama y al commit apuntados por HEAD.



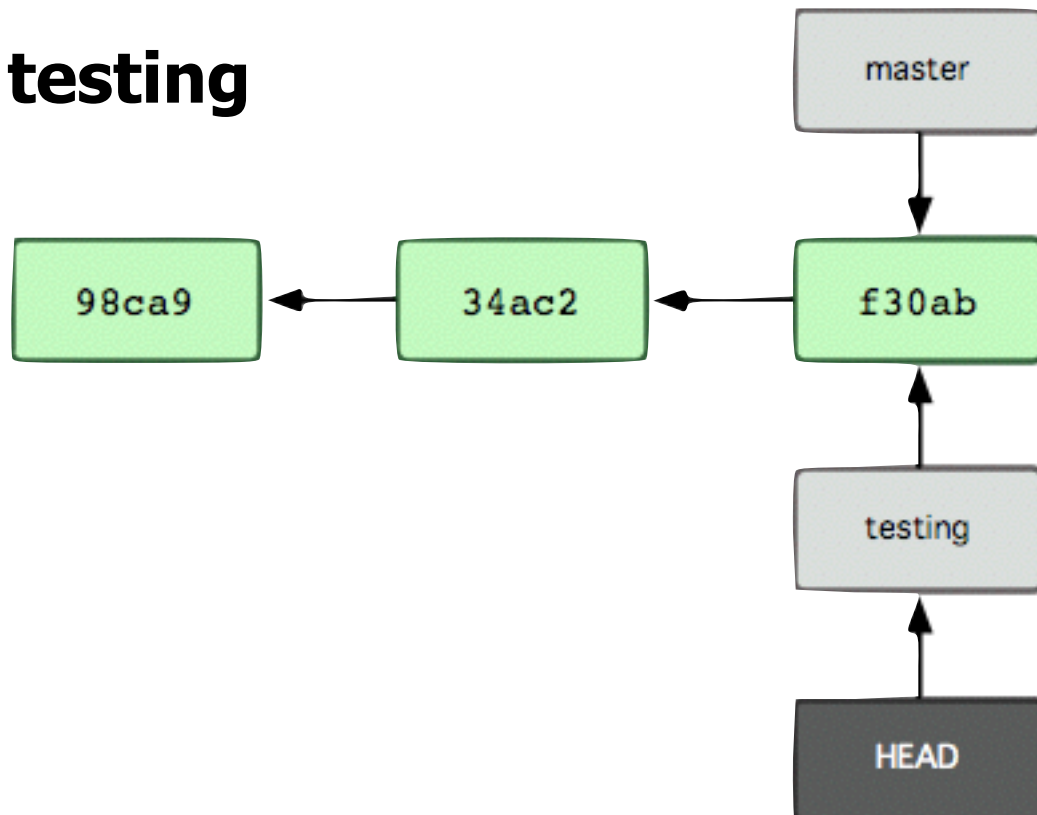
\*de Scott Chanson: <http://git-scm.org/book/>

♦ Para cambiar de rama: **git checkout** <nombre\_rama>

★ **HEAD** apuntará a la nueva rama.

★ Se actualizan el working directory y el staging area (índice)

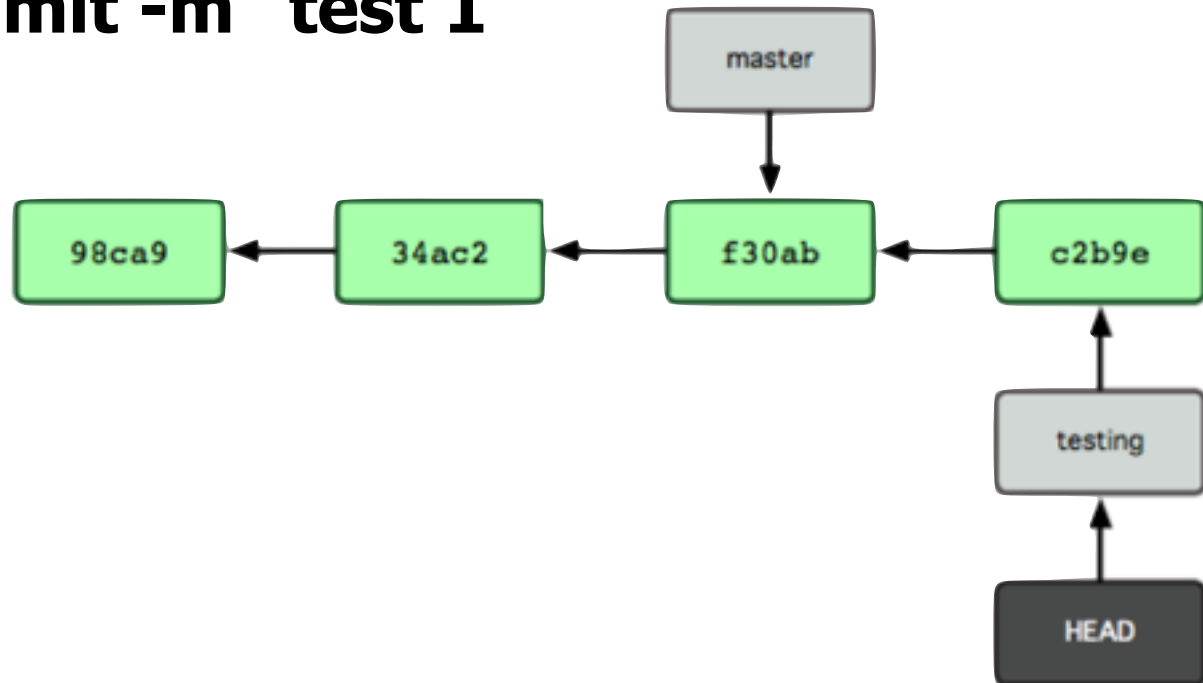
**\$ git checkout testing**



♦ Al hacer commit avanza la rama apuntada por HEAD.

**\$ git checkout testing**

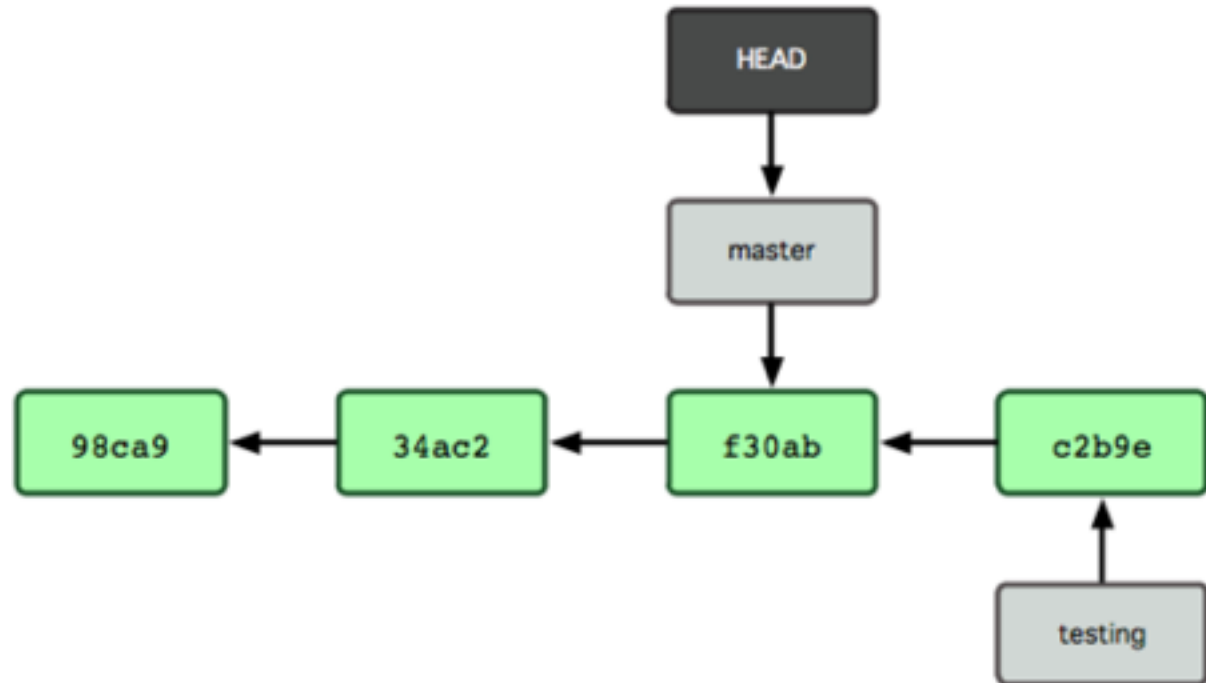
**\$ git commit -m "test 1"**





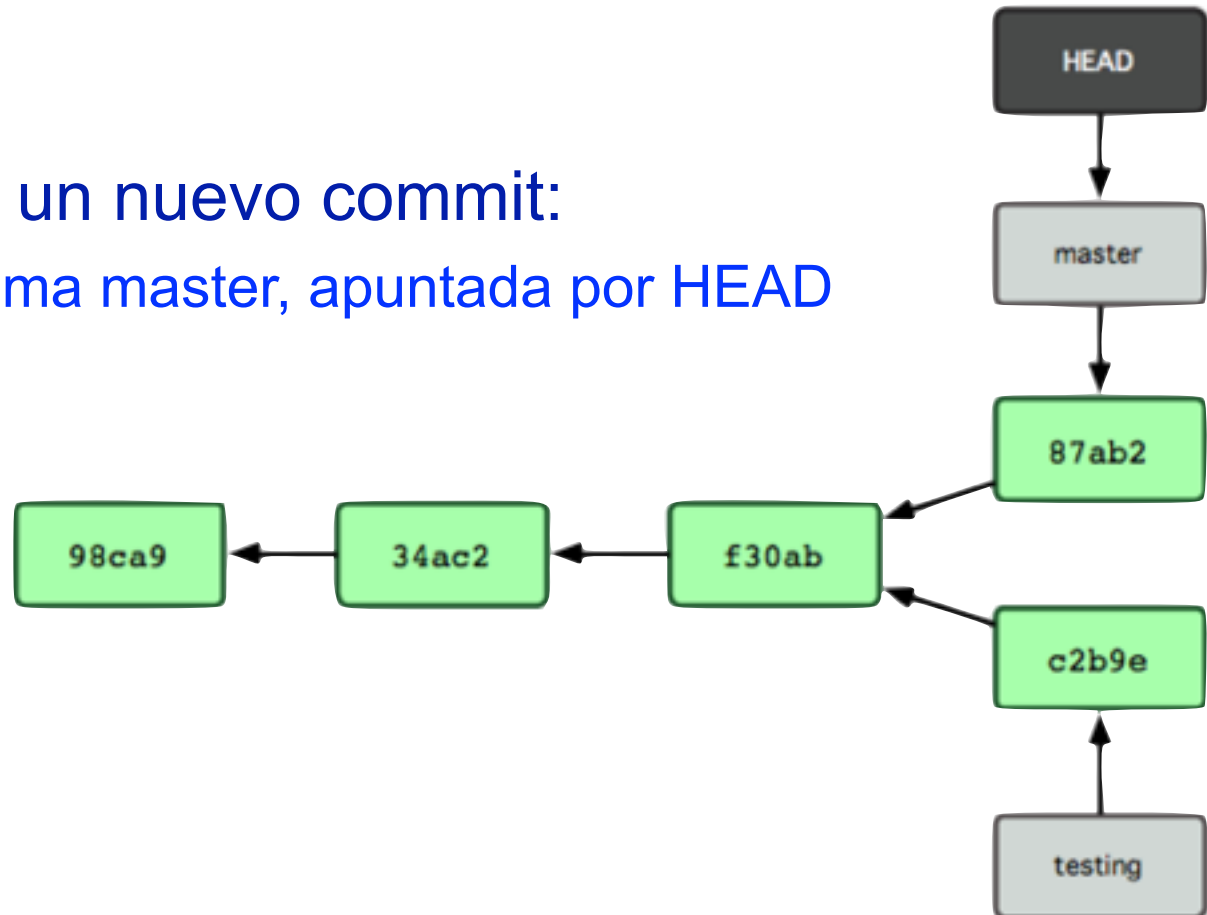
♦ Para volver a la rama master:

**\$ git checkout master**



✦ Si hacemos un nuevo commit:

★ avanza la rama master, apuntada por HEAD



# Resumen ramas

# El comando **branch** permite crear ramas

..\$ **git branch testing** # crea rama testing

..\$ **git checkout testing** # HEAD a rama testing

.....

..\$ **git add .**

..\$ **git commit -m 'test 1'** # commit de testing

..\$ **git checkout master**

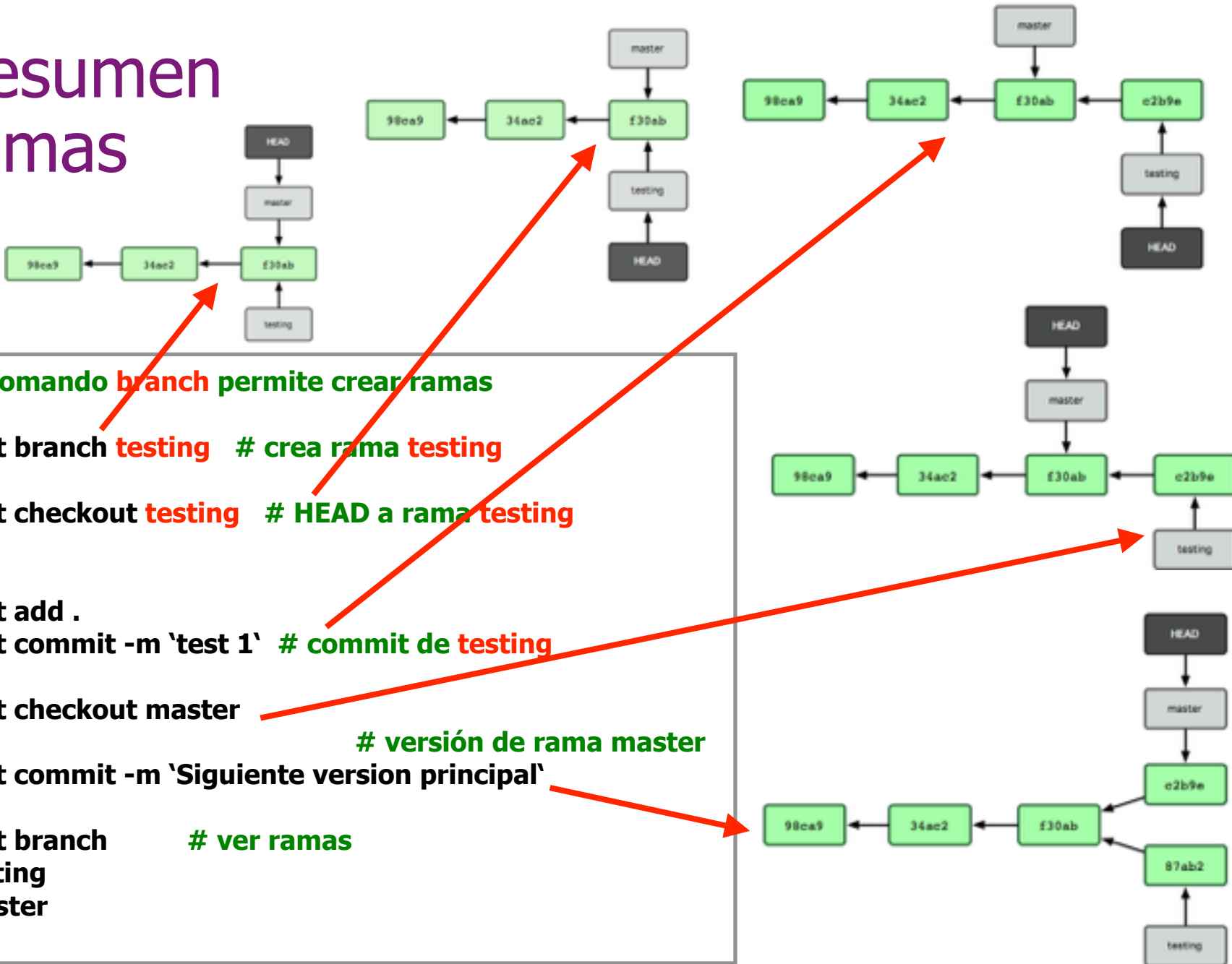
# versión de rama master

..\$ **git commit -m 'Siguiete version principal'**

..\$ **git branch testing** # ver ramas

\* master

..\$



# Crear ramas y cambiar de rama

- ✦ Para crear una rama nueva en el punto de trabajo actual:

**git branch** <rama>

- ✦ Para crear una rama nueva en un commit dado:

**git branch** <rama> <commit>

- ✦ El comando checkout con la opción -b también permite crear una rama y cambiarse a ella:

**git checkout -b** <rama> <commit>

# Cambiar de rama

- ◆ Para cambiar de rama:

**git checkout <rama>**

- ◆ Para crear una rama y cambiarse a ella:

**git checkout -b <rama>**

- ◆ Checkout falla si hay cambios en los ficheros del directorio de trabajo o en el staging area que no están incluidos en la rama a la que nos queremos cambiar.

- ★ Podemos forzar el cambio usando la opción **-f**.

- perderemos los cambios realizados

- ★ Podemos usar la opción **-m** para que nuestros cambios se mezclen con la rama que queremos sacar

- Si aparecen conflictos, los tendremos que solucionar.

# Más usos de "git checkout"

# "git checkout ." deshace cambios staged de area de trabajo

..\$ git checkout .

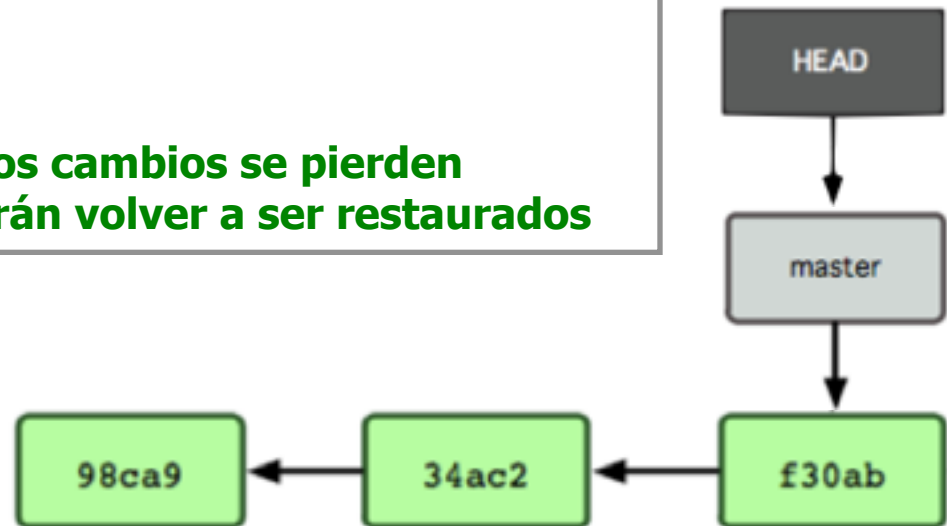
# "git checkout -- <fichero>" deshace cambios staged

# de <fichero> en area de trabajo

..\$ git checkout -- <fichero>

# **!!OJO!!** Los cambios se pierden

# y no podrán volver a ser restaurados

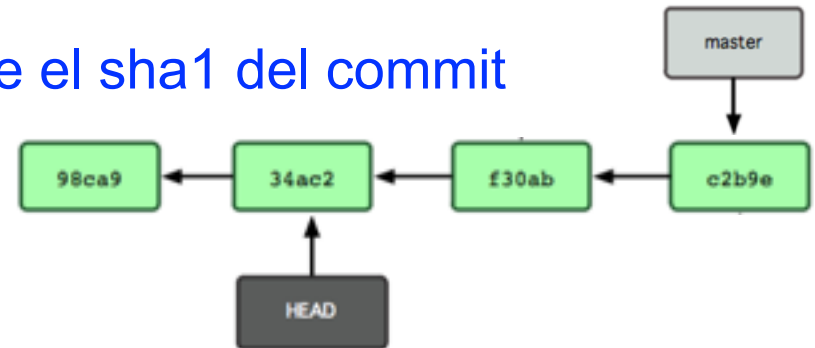


# Detached HEAD Branch

- ◆ Es una rama que se crea sobre un commit que no tiene un nombre de rama apuntándole.

**\$ git checkout 34ac2**

- ★ Se ha realizado el checkout sobre el sha1 del commit



- ◆ Suele hacerse para inspeccionar una versión anterior
- ◆ “Detached HEAD branch” crea una rama sin nombre
  - ★ Para guardar cambios debe crearse rama con: **git branch <xx>**
  - ★ También se podía haber creado la rama al realizar checkout
    - **git checkout -b <nombre\_rama> 34ac2**



# GIT

## 8. Unir ramas: merge



# Unir ramas

- ◆ Para incorporar en la rama actual los cambios realizados en otra rama:

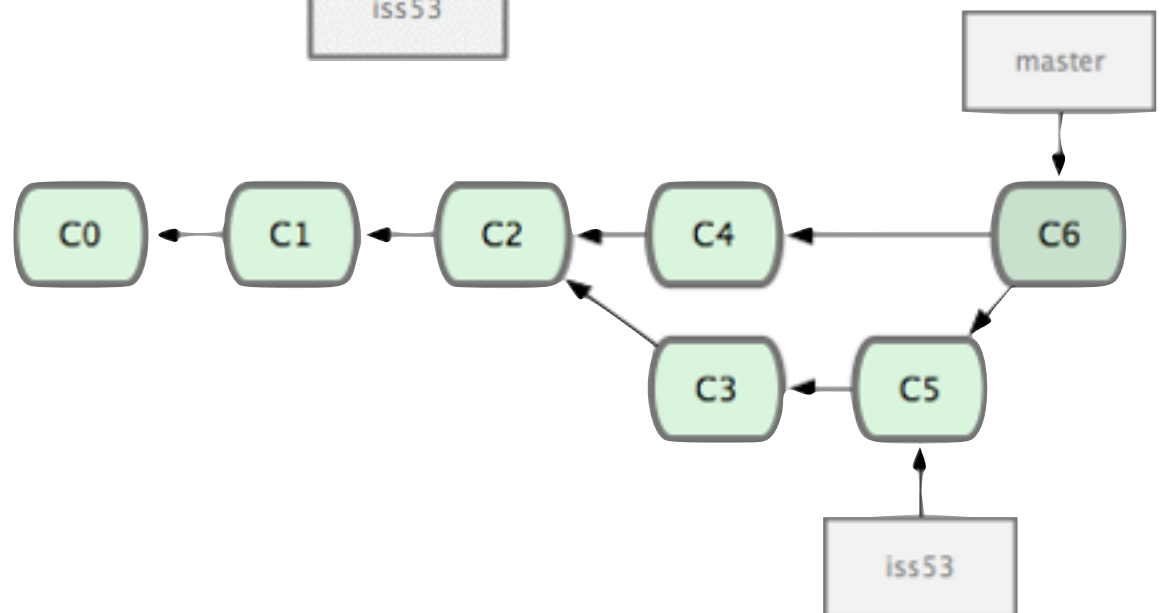
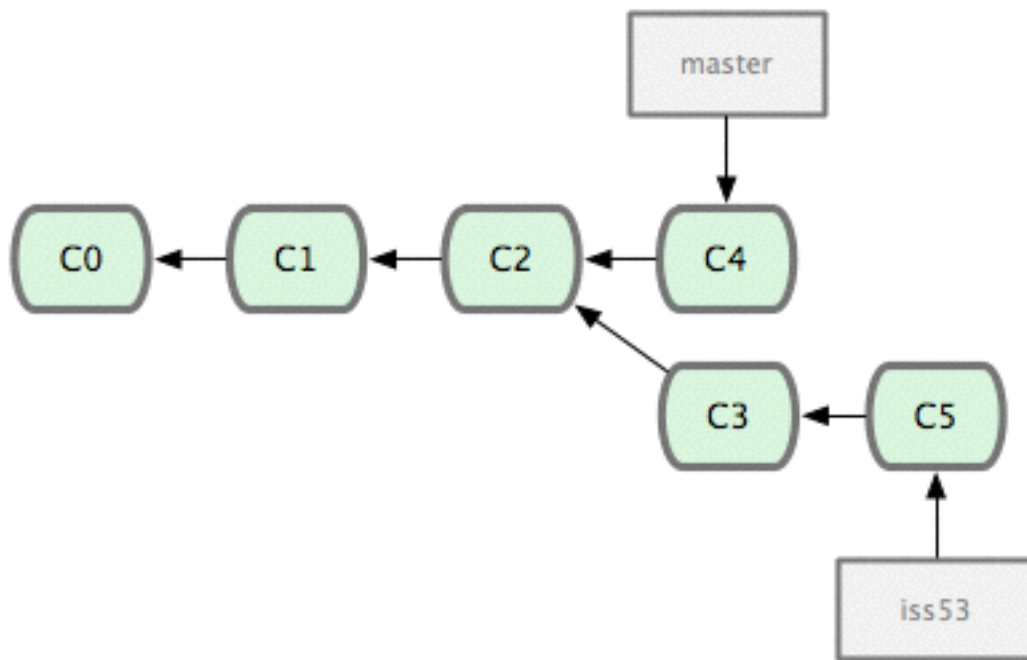
**git merge <rama>**

- ◆ Internamente GIT analiza la historia de commits para calcular como hacer la mezcla.

★ Puede hacer un fast forward, una mezcla recursiva, ...

- ◆ Ejemplo:

```
$ git checkout master # Estamos en la rama master
$ git merge iss53    # incorporamos los cambios hechos
                     # en la rama iss53 en la rama master
```



# Conflictos

- ♦ Al hacer el merge pueden aparecer conflictos
  - ★ las dos ramas han modificado las mismas líneas de un fichero.
- ♦ Si hay conflictos:
  - ★ no se realiza el commit
  - ★ las zonas conflictivas se marcan

```
<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">contact us at support@github.com</div>
>>>>>> iss53:index.html
```
  - ★ **git status** lista los ficheros con conflictos como **unmerged**.
- ♦ Para resolver el conflicto hay que:
  - ★ editar el fichero para resolver el conflicto.
  - ★ y ejecutar **git add** y **git commit**.

# Borrar ramas

- ✦ Una vez terminado el trabajo con una rama
  - ★ la borraremos con **git branch -d <nombre>**
    - Se elimina el puntero al commit.
- ✦ Si la rama a borrar no ha sido mezclada con la rama actual
  - ★ se muestra un mensaje de error y no se borra.
  - ★ Para borrar la rama independientemente de si ha sido mezclada o no, usar la opción **-D** en vez de **-d**.  
**git branch -D <nombre>**

# Listar ramas

♦ Para listar las ramas existentes: **git branch**

**\$ git branch**

**iss53**

**\* master**

**testing**

★ Se marca con un asterisco la rama activa.

♦ Opciones:

★ **-r** muestra ramas remotas

★ **-a** muestra todas las ramas (locales y remotas)

★ **-v** muestra el último commit de la rama.

★ **--merged** muestra las ramas que ya se han mezclado con la rama actual.

★ **--no-merged** muestra las ramas que aun no se han



# GIT

## 9. Repositorios remotos

# ¿Qué es un repositorio remoto?

- ◆ En un proyecto GIT real suelen existir varias copias del repositorio en servidores externos, por ejemplo
  - ★ Un repositorio en GITHUB para colaborar con otros programadores
  - ★ Un repositorio en Heroku con el que hacemos el despliegue en la nube
- ◆ Son repositorios remotos del proyecto que
  - ★ Permite subir (push) o bajar (pull, fetch) contribuciones
    - Compartiendolas con un equipo o una comunidad
- ◆ Un repositorio remoto se puede referenciar por URL o nombre
  - ★ Su URL (largo e incomodo)
  - ★ El nombre (**remote**) es más corto y más cómodo de usar

# Repositorios remotos (remote)

# El comando **git remote** lista los nombres de los remotes definidos.  
# **origin** es el remote que se creó automáticamente al clonar el proyecto.

```
$ git remote  
koke  
origin
```

# Usar la opción **-v** para ver las URL de los repositorios remotos.

```
$ git remote -v  
koke      git://github.com/koke/grit.git  
origin    git@github.com:mojombo/grit.git
```

# Para crear un remote se usa el comando **git remote add** **shortname** **URL**

```
$ git remote add pepito git://github.com/pepe/planet.git
```

# Los repositorios remotos se pueden manejar con 3 tipos de URL

```
# tipo git:      git@github.com:jquemada/swcm2012.git  
# tipo ssh:      ssh://github.com/jquemada/swcm2012  
# tipo https:    https://github.com/jquemada/swcm2012
```



# Más comandos sobre remotes

# Inspeccionar detalles de un remote: **git remote show** [nombre\_del\_remote]

# Muestra el URL del remote, información sobre las ramas remotas,  
# las ramas tracking, etc.

# Renombrar un remote: **git remote rename** nombre\_viejo nombre\_nuevo

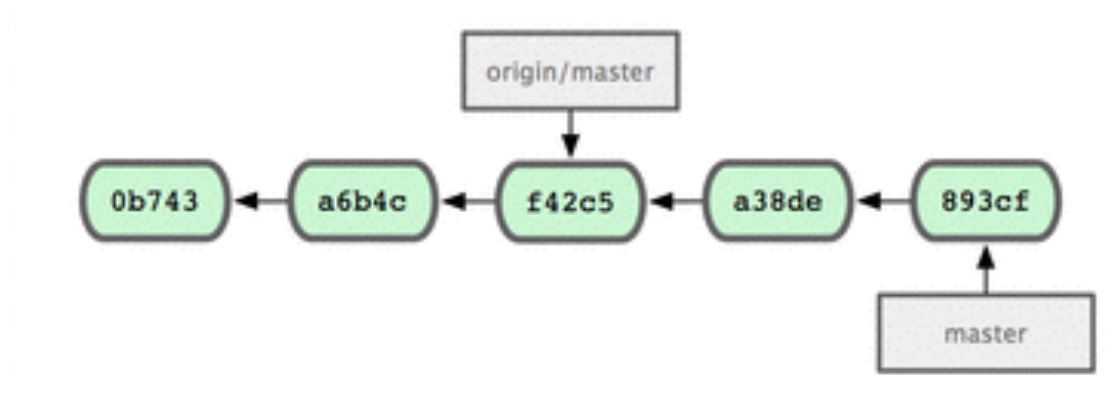
# Borrar un remote: **git remote rm** nombre\_del\_remote

# Para actualizar la información sobre los remotes definidos: **git remote update**

# Para borrar ramas que ya no existen en el remote: **git remote prune**

# Ramas Remotas

- ♦ Una rama remota es un puntero a un commit.
  - ★ Indica cual era la posición de una rama en un repositorio remoto la última vez que nos conectamos con él.
- ♦ Se nombran como: **<remote>/<rama>**.
- ♦ La figura muestra donde estaba la rama **master** en el repositorio **origin** la última vez que nos actualizamos.



- ♦ Este puntero no lo podemos mover manualmente.
  - ★ Se mueve cuando actualizamos con el repositorio remoto.

# Tracking Branch

- ◆ Es una rama local emparejada con una rama remota para que estén sincronizadas.
  - ★ Hacer un seguimiento de los cambios realizados en ellas
    - Al hacer **git push** en una tracking branch se suben los cambios locales y se actualiza la rama remota emparejada.
    - Al hacer **git pull** se actualiza la rama local con los cambios existentes de la rama remota emparejada.
- ◆ La rama master que se crea al clonar un repositorio es una tracking branch de origin/master.
- ◆ Para listar las tracking branches existentes:
  - ★ **git branch -vv**
  - ★ **git remote show <remote\_name>**

✦ Para crear una tracking branch, ejecutaremos:

**git checkout -b <branchname> <remotename>/<branchname>**

★ Se crea una rama local que hace el seguimiento de la rama remota indicada.

- Nótese que el nombre local y remoto de la rama puede ser distinto.

✦ También podemos crear una tracking branch ejecutando:

**git checkout --track <remotename>/<branchname>**

★ Se crea una rama local que hace el seguimiento de la rama remota indicada, usando el mismo nombre.

# Descargar datos de un remote

# Bajarse los datos de un remoto: **git fetch** [nombre\_del\_remote [refspec]]

# refspec:

- # - indica las ramas local y remota entre las que se hará la bajada de datos.
- # - puede ser un nombre de una rama (tanto local como remota)
- # - Si no es específica este parámetro, se bajan las actualizaciones de todas las ramas locales que también existan en el repositorio remoto.

# Este comando actualiza el repositorio con los datos existentes en el remote, pero no modifica los ficheros del directorio de trabajo.

# Las operaciones de merge las deberemos invocar explícitamente.

# Ejemplo:

# Bajarse los datos que aun no tengo del repositorio del que me cloné:

\$ **git fetch** origin

# Ahora mezclo mi rama actual con la rama demo de origin:

\$ **git merge** origin/demo

# Descargar datos y Mezclar

# Bajarse los datos de un remoto y aplicar merge:

**git pull** [nombre\_del\_remote [refspec]]

# Si la rama actual es una tracking branch:

# El comando **git pull** [nombre\_del\_remote], actualiza la rama actual con los cambios realizados en la rama asociada del repositorio remoto.

\$ **git pull origin** # Actualiza rama actual con los cambios en origin.

\$ **git pull** # Por defecto se realiza un pull de origin.

# Este comando ejecuta un fetch con los argumentos dados, y despues realiza un merge en la rama actual con los datos descargados.

\$ **git pull pepito demo** # Mezcla en la rama actual la rama demo de pepito.

# Subir datos a un remote

# De forma general, el comando para subir cambios a un remote es:

**git push** [nombre\_remote [refspec]]

# La operación push sólo puede hacerse sobre repositorios bare.

# Son repositorios donde no se desarrolla. Sólo se suben cambios.

# No tienen un directorio de trabajo.

# Los repositorios bare se crean con init o clone, y usando la opción --bare.

# Si la rama actual es una tracking branch, no suele especificarse un refspec:

#

# El comando **git push** [nombre\_del\_remote], sube los cambios desarrollados en

# la rama local actual a la rama asociada del repositorio remoto.

\$ **git push origin master** # Subir los cambios en la rama master local a origin.

\$ **git push origin** # Subir los cambios de la rama local actual o origin.

\$ **git push** # Por defecto el remote es origin.

# La operación push sólo puede realizarse si:

# - se tiene permiso de escritura en el repositorio remoto.

# - nadie ha subido nuevos cambios al repositorio remoto, es decir,  
# estamos actualizados.

# \* Si en el remote hay actualizaciones que no tenemos, deberemos hacer  
# un pull antes de poder subir nuestros cambios.

# \* No pueden subirse actualizaciones que puedan producir conflictos.

# Si no es especifica un valor para refspec, se suben las actualizaciones de todas las  
# ramas locales que también existan con el mismo nombre en el repositorio remoto.

# Si se crea una rama local, y se quiere subir al repositorio remoto, debe  
# ejecutarse el comando push con el nombre de la rama como valor de refspec:

\$ git push origin prueba

# refspec permite indicar los nombres distintos para las ramas local y remota:  
# Subir los cambios de rama local uno a la rama dos del repositorio origin:

\$ git push origin uno:dos

# refspec tiene el formato <rama\_local>:<rama\_remota>

# Para borrar una rama remota :

\$ git push origin :dos

# Usamos un refspec con un nombre de rama local vacío,  
# y con el nombre de la rama remota a borrar.





Final del tema