Entendiendo git

Nuestro blog :
www.aprendegit.com
@aprendegit

Alfonso Alba García
@aalbagarcia
info@aprendegit.com

cursodegit.com

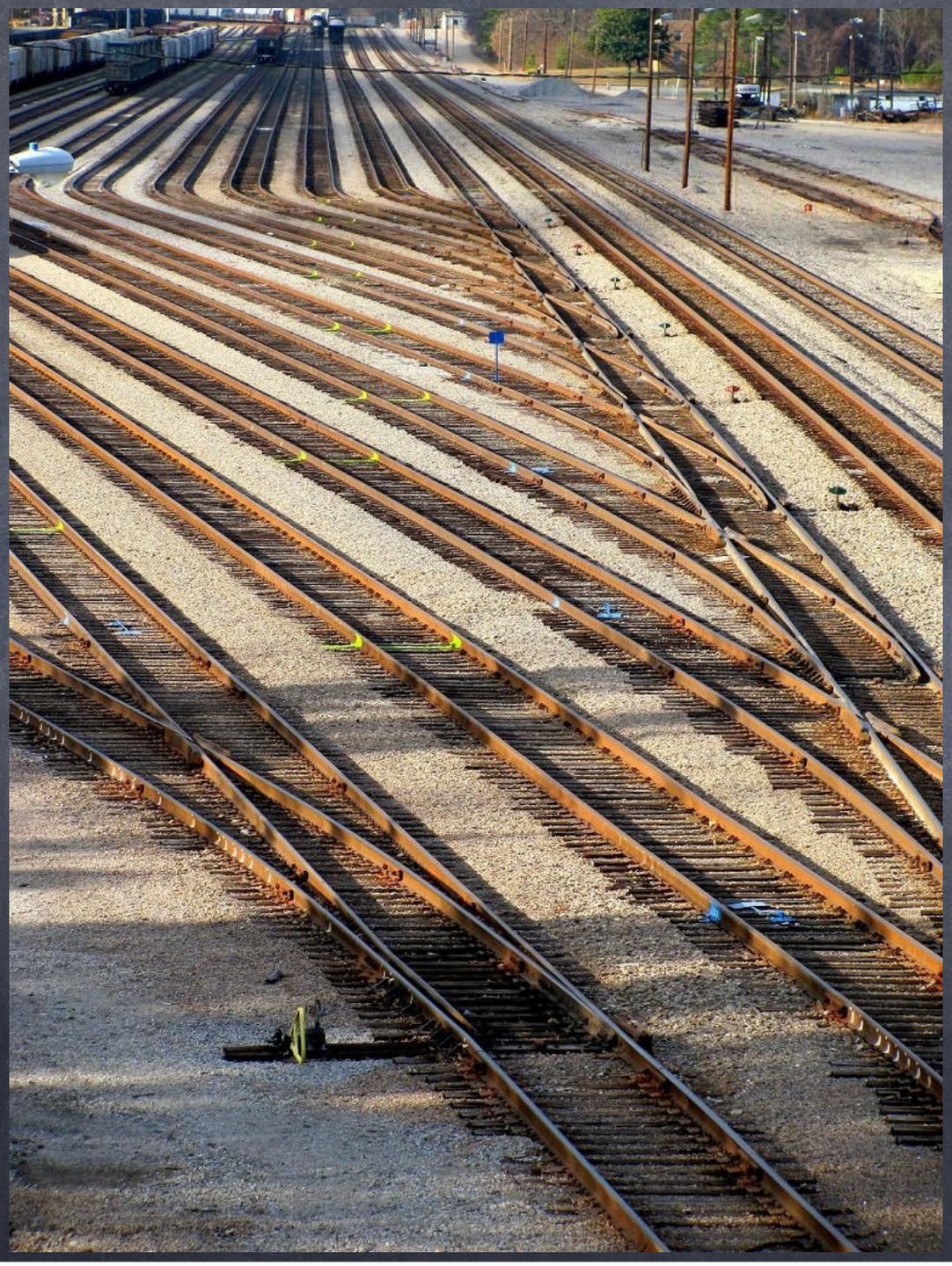
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

¿Qué es un repositorio en git?

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¡¡ESTO NO!!



cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

¿por qué estáis aquí?

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una
Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué nos permite git?

- Auditoría del código:
  - Saber qué ha pasado y cuando
  - Saber quién lo ha hecho

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué nos permite git?

- Desarrollo ágil:
  - Es sencillo crear espacios para probar cosas sin que interrumpa nuestro flujo de trabajo normal
  - Es sencillo cambiar de un tema a otro
  - Refactoring es más sencillo
  - Se integra a la perfección con la metodología TDD

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué nos permite git?

- Gestión de errores:
- Dar marcha atrás cuando nos equivocamos
- Volver a un punto anterior del desarrollo

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué nos permite git?

- Control de código:
  - Saber qué se ha entregado al cliente
  - Saber qué se ha desplegado en producción
  - Saber en qué estado está el desarrollo de una determinada característica

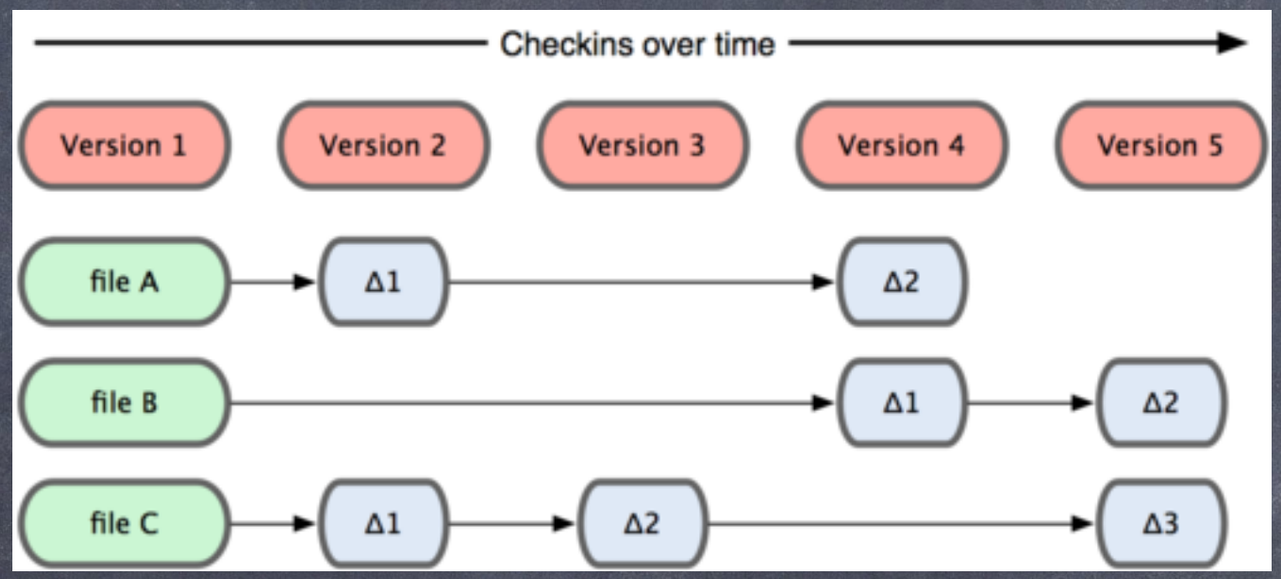
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué nos permite git?

- Trabajo en equipo:
  - Optimiza y permite el trabajo distribuido de equipos de muchas personas
- Extremadamente flexible: se implementa casi cualquier flujo de trabajo, aunque lo normal es que git te los optimice y los tengas que cambiar...

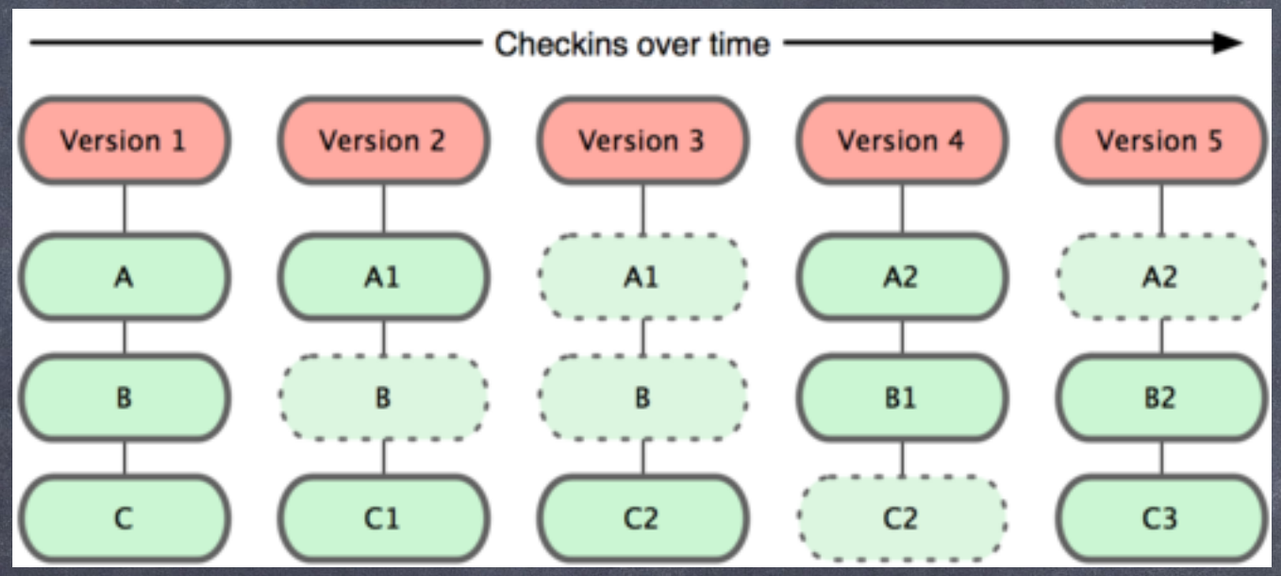
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Conceptos básicos



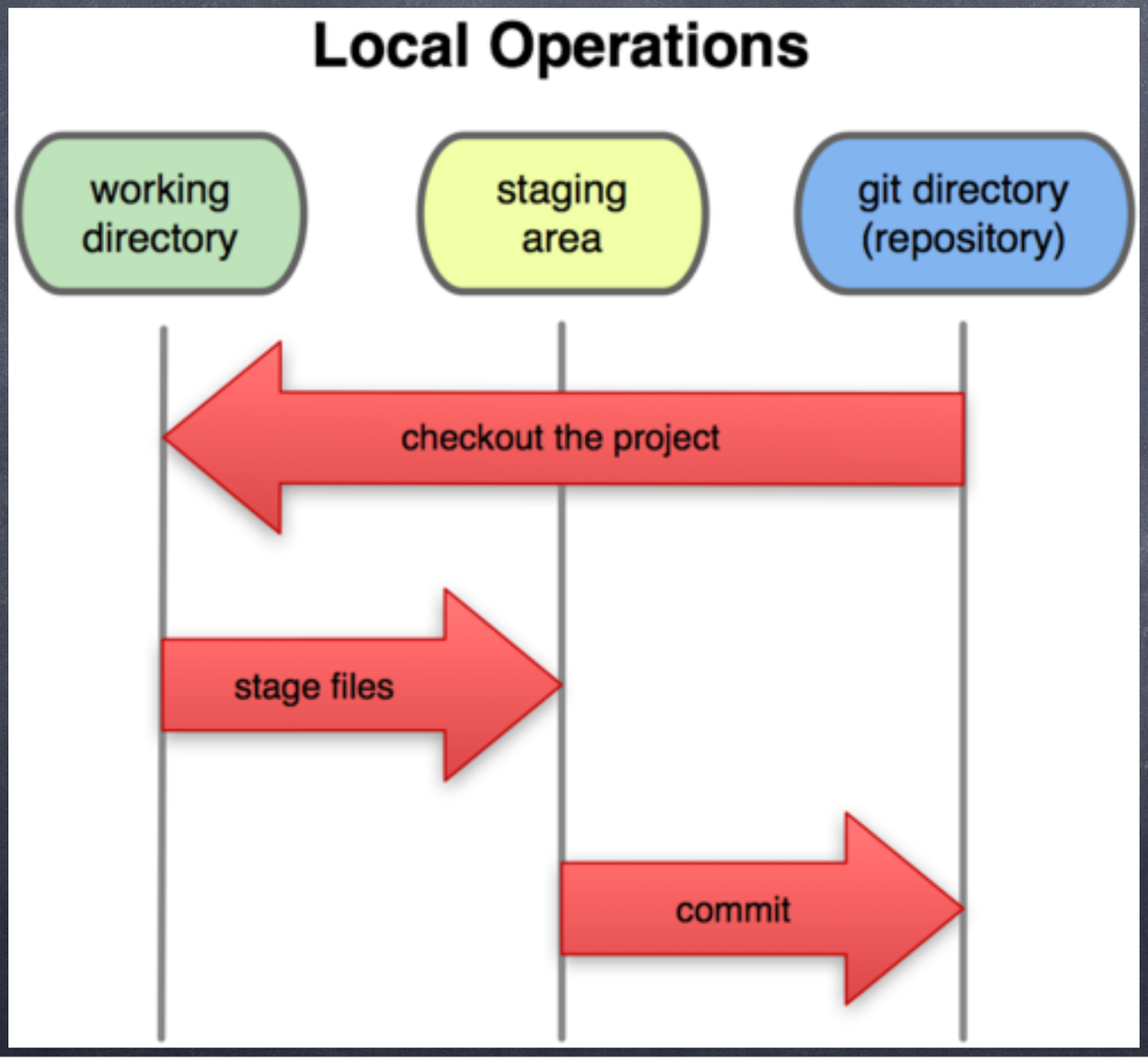
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Conceptos básicos



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Conceptos básicos



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

git sólo sabe hacer tres cosas...

- Hacer snapshots de código
- Almacenar esos snapshots en una estructura de datos (grafo)
- Gestionar referencias

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Instalación

- Linux:
  - utilizar gestor de paquetes de la distribución

    https://git-scm.com/download/linux

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Instalación

- Windows:
  - Instalar usando el instalador oficial https://git-scm.com/download/win
  - Instalar sourcetree, que incluye git una shell para poder usarlo con línea de comandos
    - Inconveniente: la versión de git que instala sourcetree no siempre es la última versión

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Instalación

- Mac:
  - Instalar usando el instalador oficial https://git-scm.com/download/mac
  - Si usas homebrew o macports, instalar el paquete correspondiente (recomendado)
  - Instalar sourcetree (ver diapositiva anterior)

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Configuración

- La configuración de git se realiza a través del comando git-config
- Existen dos tipos de configuración:
  - global: se aplica a todos los repositorios que existen en tu máquina
  - por repositorio

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Configuración

- La configuración consiste en parejas clave-valor que se almacenan por secciones en ficheros de texto:
  - global: ~/.gitconfig
  - por repositorio: ~/.git/config

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Configuración

- Sintaxis para obtener el valor de una opción

```
git config <opción>
git config --global <opción>
```

Por ejemplo: git config --global user.email

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Configuración

- Sintaxis para fijar el valor de una opción

```
git config <opción> <valor>
git config --global <opción> <valor>
```

Por ejemplo: git config user.email mi@email.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Configuración

- Podéis ver una lista completa de las opciones de configuración en https://git-scm.com/docs/git-config

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Configuración: alias

- Permite definir alias de comandos y opciones de git
- Por ejemplo, aquí tenéis un comando para mostrar un grado de git por consola:

```
git log --oneline  \
        --graph --all --decorate --color
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Configuración: alias

```
*   c07fc1a (HEAD -> master, origin/master, origin/HEAD) Merge pull request #80 from malvarezg/
|\
| * 84780a9 Commit3
|/
*   b929a9e Merge pull request #79 from juancsch/funcionalidad_molona
|\
| * 9f68e50 cambio final de la funcionalidad
| * 40858be cambio que completa el pull request
| * 271d7d1 Correcion de un bug en la funcionalidad super molona
| * 4a06a91 Inicio de la nueva funcionalidad super molona
|/
*   472dd0a Merge pull request #77 from diegogplfree/pagos-1
|\
| * 2cfeecb nuevo commit
| * 81396a5 pagos-1
| * 8bb5cc4 pagos-1
| * 47842e9 new file darboleda
|/
*   bd1533c Merge pull request #71 from LoadCode/mycommits
|\
| * 0a0165f Una tercera linea agregada al archivo HelloWorld para completar la pull request con
| * 9a3b565 Agregada una linea del Hola Mundo con Ruby
| * 28d5118 Agregado un archivo HelloWorld en Ruby
|/
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Configuración: alias

- Definimos un alias para no tener que acordarnos de todas las opciones

```
git config --global alias.gr 'log --oneline
  --graph --all --decorate --color'
```

A partir de este momento podemos usar simplente:

```
git gr
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Configuración

- Personalizando git: https://git-scm.com/book/en/v2/Customizing-Git-Git-Configuration
- Aliases: https://git-scm.com/book/en/v2/Git-Basics-Git-Aliases
- https://git-scm.com/book/en/v2/Git-Commands-Setup-and-Config
- https://git-scm.com/docs/git-config

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Estructura del repositorio local

- ¿Qué podemos encontrar en la carpeta .git?
  - .git/objects

en esta carpeta se almacenan los objetos de git: commits, trees, blobs, etc

por ejemplo, si un commit tiene un hash ab123dc, encontraremos el commit en la carpeta .git/ab/123dc...

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Estructura del repositorio local

- .git/refs
  Contiene las referencias de nuestro repositorio: branches, tags y remote branches

por ejemplo, si creamos una rama "features/nuevos_servicios" que apunta al commit ab12ca2..., se creará un fichero de texto en .git/refs/heads/features/nuevos_servicios con contiene ese hash

Ejemplo: crea un repositorio git con un commit y mira el contenido de .git/refs/heads/master

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Estructura del repositorio local

- .git/refs
  Dentro de .git/refs/tags encontraremos las etiquetas

  Si creamos una etiqueta v1.0.0 "lightweight" encontraremos un fichero en .git/refs/tags/v1.0.0 con el hash del commit al que apunta la etiqueta

  Si creamos una etiqueta anotada v1.0.0, se crea el mismo fichero .git/refs/tags/v1.0.0, pero este contiene un hash al objeto git que contendrá el hash del commit etiquetado junto con el resto de la información (comentario, firma, etc)

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Estructura del repositorio local

- .git/config

  Fichero que contiene la configuración del repositorio, por ejemplo:

  - Ramas, ramas remotas y tracking branches
  - Remotos
  - Nombre del autor e email para ese repositorio

- Este fichero de configuración no se comparte, es decir, no se sube al repositorio remote cuando haces push.

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Estructura del repositorio local

- .git/HEAD

  Puntero a la rama activa.

- .git/index

  Fichero binario que contiene el índice de git, también conocido como staging area o "cache"

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Estructura del repositorio local

- .git/hooks

Contiene los script que se ejecutan durante el ciclo de vida de los commits (hooks)

Más información sobre hooks:
Pro git - Puntos de enganche

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Estructura del repositorio local

- .git/logs

Almacena los logs de las referencias (ramas). Usar el comando reflog para acceder a los logs.

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

.gitignore

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

.gitignore

- Tres formas de ignorar ficheros (por orden de prioridad)
  - .git/info/exclude -> sólo este repositorio
  - .gitignore -> todos los que clonan el repositorio
  - .gitignore_local -> todos mis repositorios locales
    - git config --get core.excludesfile
- Se aplica el último patrón que se lee

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

.gitignore: Patrones

- Líneas en blanco: se ignoran
- # Comentarios
- ! negación
- Patrones terminados en / se interpretan como carpetas
- Si el patrón empieza por /, se interpreta como ruta relativa
- El resto de patrones se interpretan como Shell globs

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

.gitignore: Patrones

- * -> Comodín (no incluye el separador de directorios)
  - el patrón web/*.html ignora el fichero web/index.html pero no ignora web/blog/index.html
- [ ] -> Selección.
- ? -> cualquier caracter (sólo uno)

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

.gitignore: ejemplos

- Ejemplos de patrones:

```
.idea/
*.html
web/*.html
/*.php
*.[xn]ib
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

.gitignore

- Más información:
  - Página de manual de .gitignore
  - Serie de artículos sobre .gitignore en aprendegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El fichero .gitignore para proyectos XCode

- http://stackoverflow.com/questions/49478/git-ignore-file-for-xcode-projects
- https://gist.github.com/adamgit/3786883

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Creación de un repositorio

Crea una carpeta, entra dentro y ejecuta el comando

```
git init
```

el repositorio se inicializa y aparece la carpeta .git

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Estado del repositorio

¿Cómo están los ficheros en los que estamos trabajando?

```
git status
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Estado del repositorio

```
$ git status
# On branch lugares
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#  modified:   app/controllers/backend/application_controller.rb
#  modified:   config/routes.rb
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   app/assets/javascripts/backend/cities.js.coffee
#   app/assets/javascripts/backend/resource.js.coffee
#   app/assets/stylesheets/backend/cities.css.scss
#   app/assets/stylesheets/backend/resource.css.scss
#   app/controllers/backend/cities_controller.rb
#   app/controllers/backend/resource_controller.rb
#   app/helpers/backend/cities_helper.rb
#   app/helpers/backend/resource_helper.rb
#   app/views/backend/
#   app/views/layouts/backend.html.erb
#   test/controllers/backend/cities_controller_test.rb
#   test/controllers/backend/resource_controller_test.rb
#   test/helpers/backend/cities_helper_test.rb
#   test/helpers/backend/resource_helper_test.rb
no changes added to commit (use "git add" and/or "git commit -a")
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué ocurre cuando se crea un commit?

- Se crea un snapshot del código fuente completo (exceptuando aquellos ficheros ignorados)
- A ese snapshot se le añade información: nombre del autor, email, fecha, comentario, referencia al commit anterior
- Se calcula el la suma SHA del snapshot más la información
- La rama activa (aquella a la que apunta HEAD) pasa a apuntar al nuevo commit

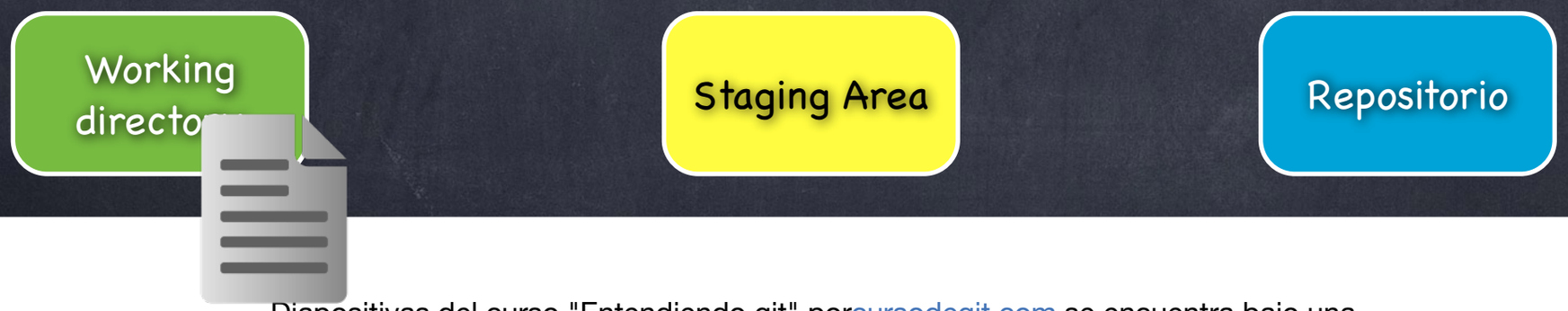
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

Añadimos ficheros al Staging Area

```
git add *
```

```
git add <fichero> <fichero>
```



cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

Añadiendo ficheros

Añadimos ficheros al Staging Area

```
git add *
```

```
git add <fichero> <fichero>
```

Working directory

Staging Area



Repositorio

cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

Añadiendo ficheros

Se puede iniciar un proceso interactivo para seleccionar qué partes de un fichero queremos subir al staging area. Ejecutamos

git add -i

```
$ git add -i
           staged     unstaged path
  1:    unchanged        +2/-2 index.html

*** Commands ***
  1: [s]tatus  2: [u]pdate  3: [r]evert  4: [a]dd untracked
  5: [p]atch   6: [d]iff    7: [q]uit    8: [h]elp
What now>
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

Seleccionar la opción "p" y posteriormente el fichero seleccionando el número que nos muestra la interfaz

```
$ git add -i
           staged     unstaged path
  1:    unchanged        +6/-2 index.html

*** Commands ***
  1: [s]tatus  2: [u]pdate  3: [r]evert  4: [a]dd untracked
  5: [p]atch   6: [d]iff    7: [q]uit    8: [h]elp
What now> p
           staged     unstaged path
  1:    unchanged        +6/-2 [i]ndex.html
Patch update>> 1
           staged     unstaged path
* 1:    unchanged        +6/-2 [i]ndex.html
Patch update>>
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

En este punto, presionamos de nuevo "Intro" y entramos en el modo interactivo

cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

```
$ git add -i
           staged     unstaged path
  1:    unchanged        +6/-2 index.html

*** Commands ***
  1: [s]tatus  2: [u]pdate  3: [r]evert  4: [a]dd untracked
  5: [p]atch   6: [d]iff    7: [q]uit    8: [h]elp
What now> p
           staged     unstaged path
  1:    unchanged        +6/-2 [i]ndex.html
Patch update>> 1
           staged     unstaged path
* 1:    unchanged        +6/-2 [i]ndex.html
Patch update>>
diff --git a/index.html b/index.html
index b08ce9a..3c50d0c 100644
--- a/index.html
+++ b/index.html
@@ -21,8 +21,8 @@
       <li><a href="#">Github</a></li>
       <li><a href="#">XCode</a></li>
       <li><a href="#">¿Quiénes somos?</a></li>
-      <li><a href="#">Cursos</a></li>
-      <li><a href="#">Contacto</a></li>
+      <li><a href="#">Cursos de git</a></li>
+      <li><a href="#">Contacta con nosotros</a></li>
     </ul>
     </div>
   </div>
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]?
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

Puedes seleccionar qué hacer con los pedazos (hunks) de código que has modificado en el fichero

```
$ git add -i
...
Patch update>>
diff --git a/index.html b/index.html
index b08ce9a..3c50d0c 100644
--- a/index.html
+++ b/index.html
@@ -21,8 +21,8 @@
        <li><a href="#">Github</a></li>
        <li><a href="#">XCode</a></li>
        <li><a href="#">¿Quiénes somos?</a></li>
-       <li><a href="#">Cursos</a></li>
-       <li><a href="#">Contacto</a></li>
+       <li><a href="#">Cursos de git</a></li>
+       <li><a href="#">Contacta con nosotros</a></li>
      </ul>
      </div>
    </div>
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]?
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

Para saber qué es cada una de las opciones, pulsa ?
y luego "Intro"

```
$ git add -i
...
Patch update>>
diff --git a/index.html b/index.html
index b08ce9a..3c50d0c 100644
--- a/index.html
+++ b/index.html
@@ -21,8 +21,8 @@
        <li><a href="#">Github</a></li>
        <li><a href="#">XCode</a></li>
        <li><a href="#">¿Quiénes somos?</a></li>
-       <li><a href="#">Cursos</a></li>
-       <li><a href="#">Contacto</a></li>
+       <li><a href="#">Cursos de git</a></li>
+       <li><a href="#">Contacta con nosotros</a></li>
      </ul>
      </div>
    </div>
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? ?
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

```
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? ?
y - stage this hunk
n - do not stage this hunk
q - quit; do not stage this hunk nor any of the remaining ones
a - stage this hunk and all later hunks in the file
d - do not stage this hunk nor any of the later hunks in the file
g - select a hunk to go to
/ - search for a hunk matching the given regex
j - leave this hunk undecided, see next undecided hunk
J - leave this hunk undecided, see next hunk
k - leave this hunk undecided, see previous undecided hunk
K - leave this hunk undecided, see previous hunk
s - split the current hunk into smaller hunks
e - manually edit the current hunk
? - print help
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

```
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? ?
y - Pasar este trozo al área de ensayo
n - No pasar este trozo
q - terminar. No pasa este trozo ni ninguno de los que quedan al staging area
a - Pasa este pedazo y todos los que quedan al staging area
d - No pases este trozo ni ninguno de los que quedan en el fichero al staging area
g - Selecciona un trozo para situarnos en el
/ - Busca un trozo que encaje con una expresión regular
j - Mantener este trozo como “indeciso” y ve al siguiente hunk marcado como “indeciso”
J - Mantener este trozo como “indeciso” y ve al siguiente
k - Mantener este trozo como “indeciso” y ve al siguiente hunk marcado como “indeciso”
K - siguiente hunk marcado como “indeciso” y ve al anterior
s - rompe este trozo en trozos más pequeños
e - editar este trozo manualmente
? - ayuda
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

Al terminar, podemos usar la opción "s" para ver el estado y posteriormente "q" para terminar

Cambios subidos al staging area

Cambios fuera del staging area ¡no estarán en el siguiente commit!

```
What now> s
           staged     unstaged path
  1:        +2/-2        +4/-0 index.html

*** Commands ***
  1: [s]tatus  2: [u]pdate  3: [r]evert  4: [a]dd untracked
  5: [p]atch   6: [d]iff    7: [q]uit    8: [h]elp
What now> q
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadir ficheros

Más información:

Sección 6.2 - Pro git

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

Pasamos los ficheros al repositorio

```
git commit -m'Mensaje del commit'
```

Repositorio

Utilizad siempre un mensaje corto y descriptivo
Evitad mensajes como "trabajando" "..." que no aportan nada al que lo lee

Working directory

Staging Area

Repositorio



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

Si nos hemos olvidado algún fichero al hacer el último commit, podemos añadirlo al staging area con

```
git add <fichero>
```

y modificar el commit ejecutando

```
git commit --amend
```

La opción --amend también permite modificar el comentario del commit

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Añadiendo ficheros

Recuerda que

```
git commit --amend
```

va a sobreescribir la historia del proyecto

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Borrando y moviendo ficheros

Borrar un fichero del índice

```
git rm <fichero>
```

Mover un fichero

```
git mv <ruta actual> <ruta nueva>
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Etiquetando los commits

Etiquetamos los commits para poder recuperar el código fuente en un estado determinado en el futuro

Etiqutar el último commit de la rama activa

```
git tag <nombre> –m’Mensaje’
```

Etiquetar un commit determinado

```
git tag <nombre> –m’Mensaje’ <commit>
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Etiquetando los commits

Borrar una etiqueta

```
git tag -d <nombre>
```

Ver todas las etiquetas

```
git tag -l
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Etiquetando los commits

- Tipos de etiquetas
  - Etiquetas ligeras: sólo contienen el hash del commit al que apuntan
  - Etiquetas anotadas: permiten almacenar un comentario junto al hash del commit al que apuntan

    Se crean con la opción -a

```
git tag -d <nombre> -m ‘comentario’
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Etiquetando los commits

- Tipos de etiquetas
  - Etiquetas firmadas: permite firmar las etiquetas con GPG

  Se crean con la opción -s

```
git tag -s <nombre> -m ‘comentario’
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Etiquetando los commits

- Para subir las etiquetas al repositorio remoto

```
git push --tags
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Sacando ficheros del Staging Area

Deshacer los cambios de un fichero que todavía no está en Staging Area

```
git checkout -- <fichero>
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Sacando ficheros del Staging Area

Sacar un fichero que está en el Staging Area

```
git reset HEAD <fichero>
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Sacando ficheros del Staging Area

Sacar un fichero que está en el Staging Area

```
git reset HEAD <fichero>
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Sacando ficheros del Staging Area

Sacar todos los ficheros que están en el Staging Area

```
git reset HEAD
```



cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

Ramas

Una rama en git es una etiqueta

Creación de una rama en el último commit de la rama activa

```
git branch <nombre>
```

Crear una rama en un commit

```
git branch <nombre> <commit>
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Ramas

Crear una rama y activarla

```
git checkout -b <nombre>
```

Cambiar de rama

```
git checkout <nombre>
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una
Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Ramas

Para incorporar los cambios de `feature_x` en `master` cambiamos a la rama master

```
git checkout master
```

e incorporamos los cambios de la rama `feature_x`

```
git merge feature_x
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Ramas: rebase

Para incorporar los cambios de `feature_x` en `master` cambiamos a la rama `feature_x`

```
git checkout feature_x
```

e incorporamos los cambios a la rama `master`

```
git rebase master
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando checkout

- Este comando tiene varios usos:
  - cambiar de rama, extrayendo al área de trabajo el snapshot del commit al que apunta la rama a la que cambiamos
  - crear un rama y hacer checkout de esa rama
  - para extraer el snapshot completo de un commit y ponerlo en el área de trabajo
  - extraer ficheros de otros commits y ponerlos en área de trabajo
  - descartar cambios en el área de trabajo

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando checkout

- Cambiar de rama

git checkout <nombre de la rama>

Esta es una operación segura, si para cambiar de rama git tuviese que descartar los cambios en el área de trabajo, la operación no se realiza

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando checkout

- Crear una rama y hacer checkout de esa rama

git checkout -b <rama> <dónde la creamos>

Si no le decimos dónde la queremos crear, la crea en HEAD

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando checkout

- Extraer al área de trabajo un commit, un tag o una rama remota (quedando en estado "detached HEAD")

```
git checkout <SHA del commit>
git checkout <nombre del tag o rama remota>
```

Esta es una operación segura

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando checkout

- extraer ficheros de otros commits y ponerlos en área de trabajo

git checkout <referencia> <rutas...>

Por ejemplo:
git checkout v1.1.0 app/views/the_view.html

Este comando extrae el fichero app/views/the_view.html tal cuál estaba en el tag (o rama) v1.1.0 y lo pone en el área de trabajo

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando checkout

- descartar cambios en el área de trabajo

Supongamos que hemos modificado el fichero index.html. Por el motivo que sea, queremos descartas estas modificaciones:

git checkout index.html

OJO: una vez descartados los cambios, estos no se pueden recuperar

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando checkout

- https://git-scm.com/docs/git-checkout

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

- La operación merge consiste en elaboración de un nuevo commit, denominado "merge commit", a través de la mezcla del código dos commits
- Como todos los commits, se crea siguiendo el mismo proceso:
  - Se genera un nuevo snapshot, mezcla de los snapshots de los commits que se quieren "juntar"
  - Se le añade la información del commit, que incluye dos referencias a los dos commits que se mezclaron
  - Se mueve el puntero a la rama activa (HEAD)

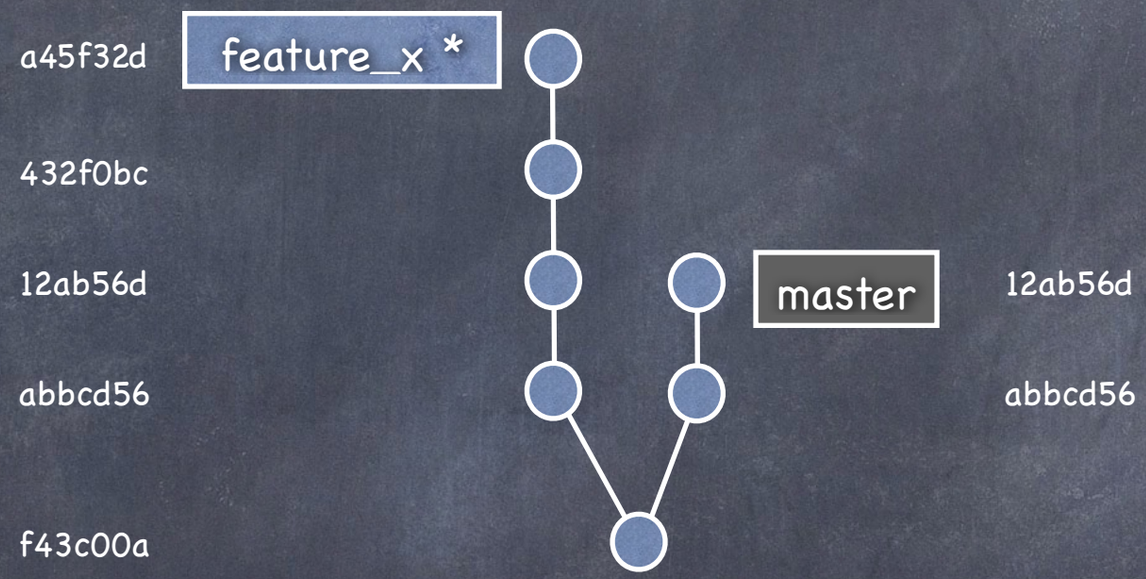
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

- Ejemplo: estamos desarrollando una nueva funcionalidad en la rama feature_x
- Según estamos desarrollando, nos damos cuenta de que la rama master ha sido modificada
- El objetivo: mezclar mi código (en la rama feature_x) con el que está en master para seguir trabajando en la funcionalidad

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge



cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

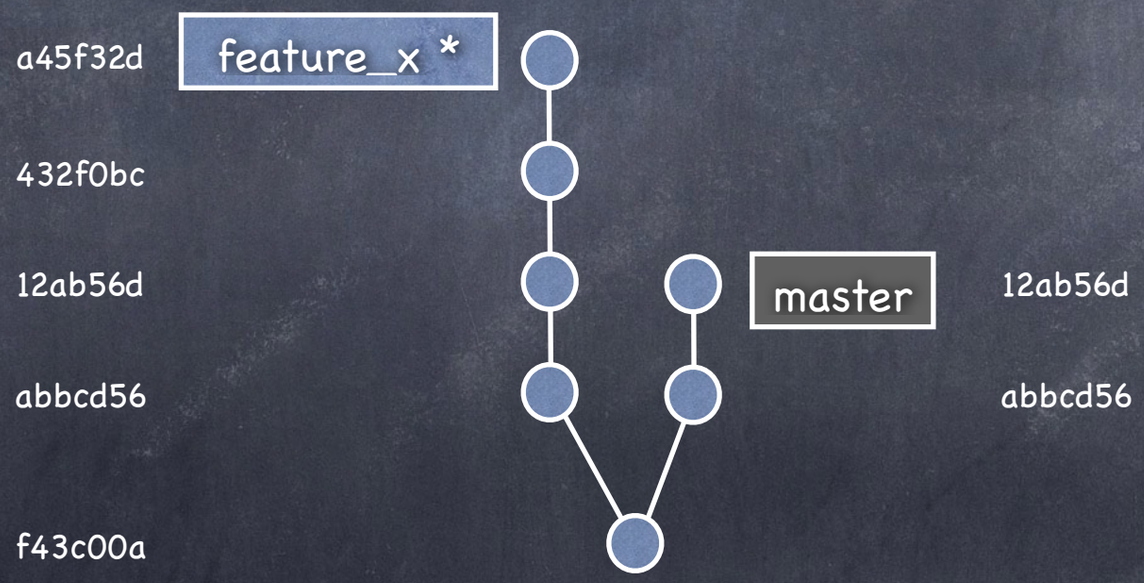
merge

- Procedimiento:
  - Usamos la rama feature_x para hacer la integración.
  - Vamos a suponer que el trabajo en feature_x **NO** está terminado, es decir, tenemos que seguir trabajando en la funcionalidad
  - "Nos vamos a " la rama feature_x (haciendo un git-checkout)
  - "Cogemos" el último commit de la rama master con el comando git-merge

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

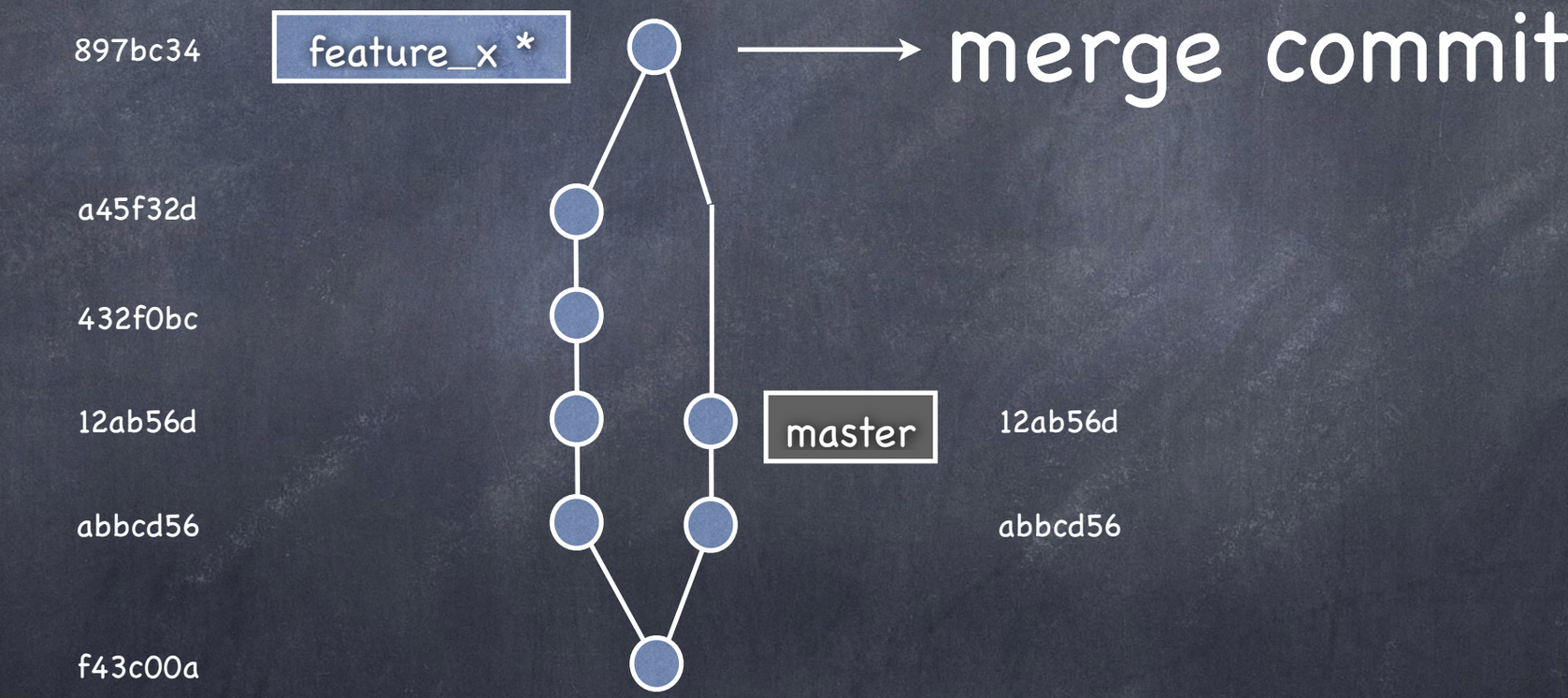
```
git checkout feature_x
  git merge master
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

```
git checkout feature_x
git merge master
```



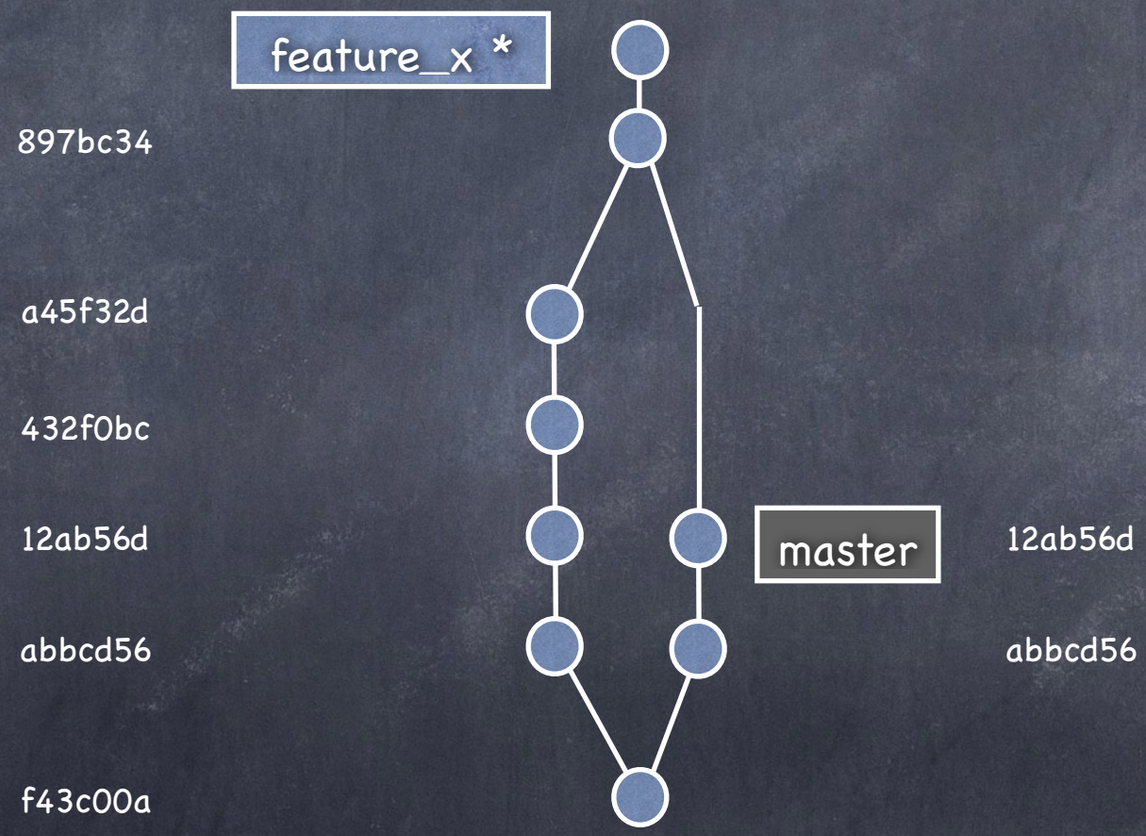
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

- Objetivo cumplido: ahora tengo la mezcla de los dos códigos y puedo seguir trabajando (haciendo commits)

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

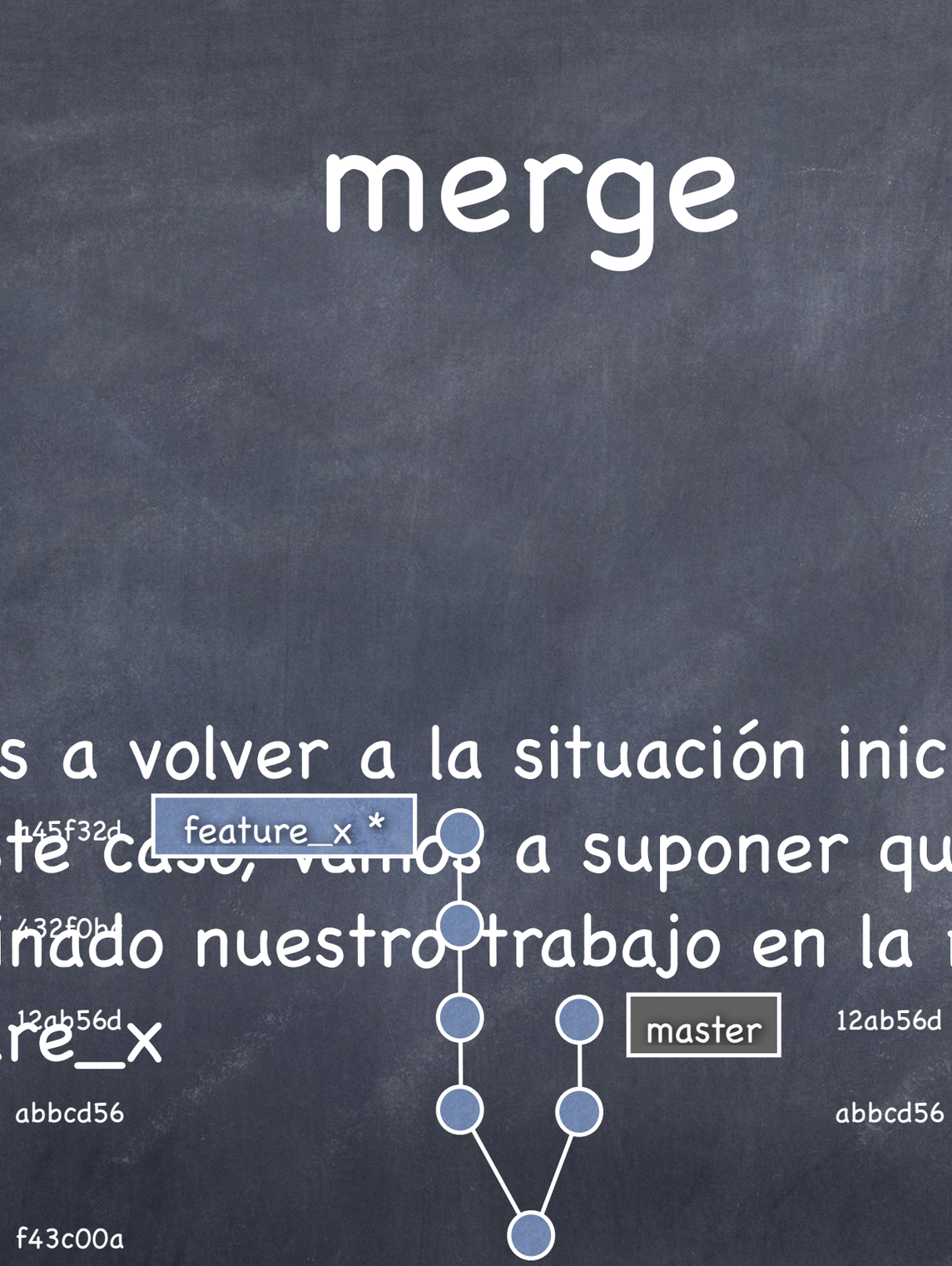
merge



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

- Vamos a volver a la situación inicial, aunque en este caso, vamos a suponer que SI hemos terminado nuestro trabajo en la rama feature_x



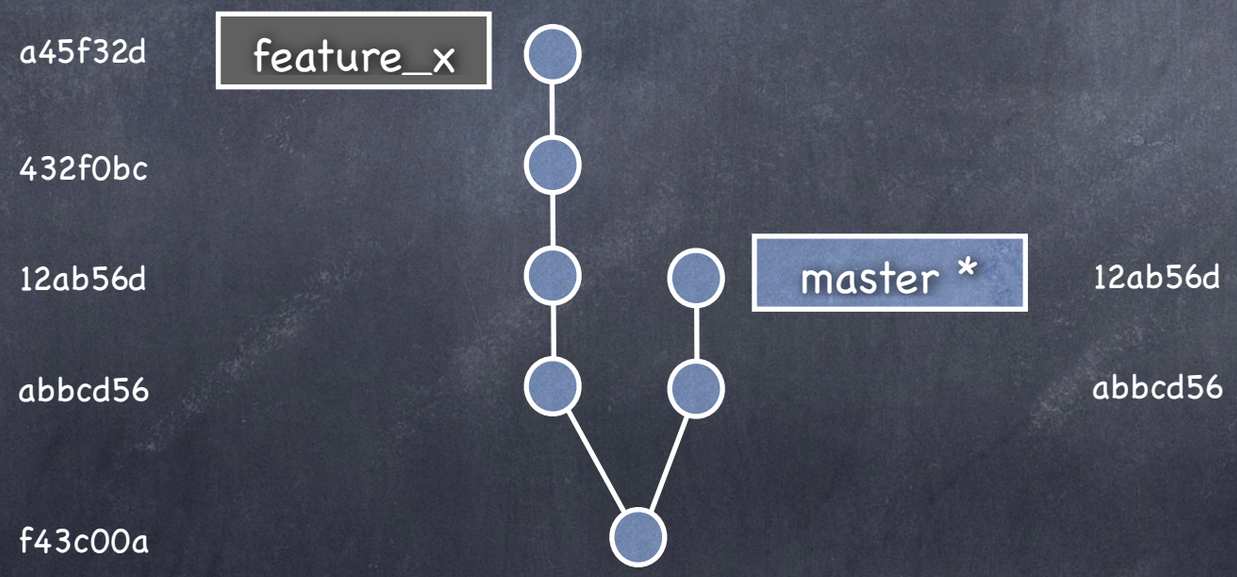
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

- En este caso, lo que queremos es poner en master ya directamente la mezcla de los dos commits:
- Usamos la rama master para hacer la integración (despliegue).
- "Nos vamos a" la rama master (haciendo un git-checkout)
- "Cogemos" el último commit de la rama feature_x con el comando git-merge

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

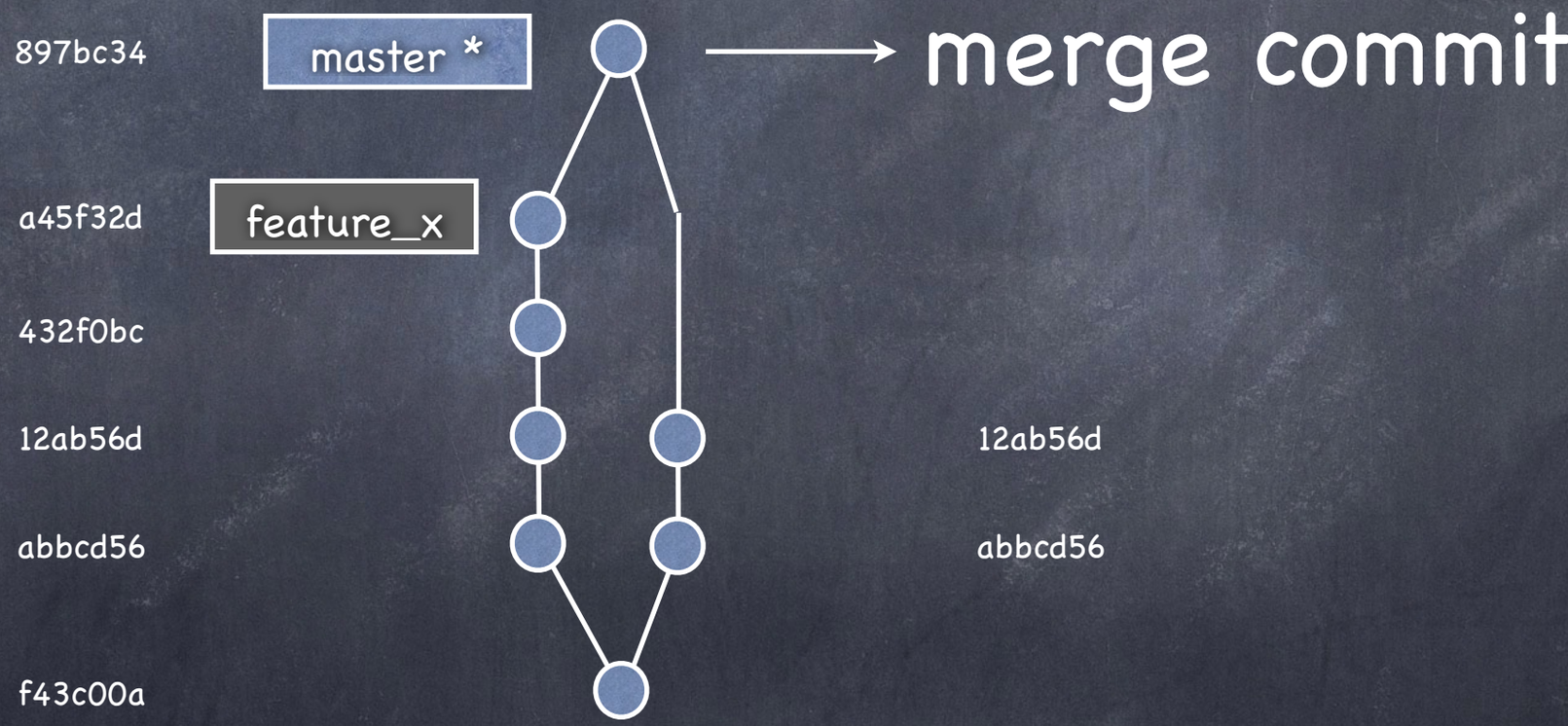
merge



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

```
git checkout master
git merge feature_x
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

- ¿hay alguna diferencia?
  - Si lo hacemos de esta manera, el código queda directamente mezclado en master
- El snapshot de código es el mismo
- La información del commit cambia: el orden de los padres es diferente

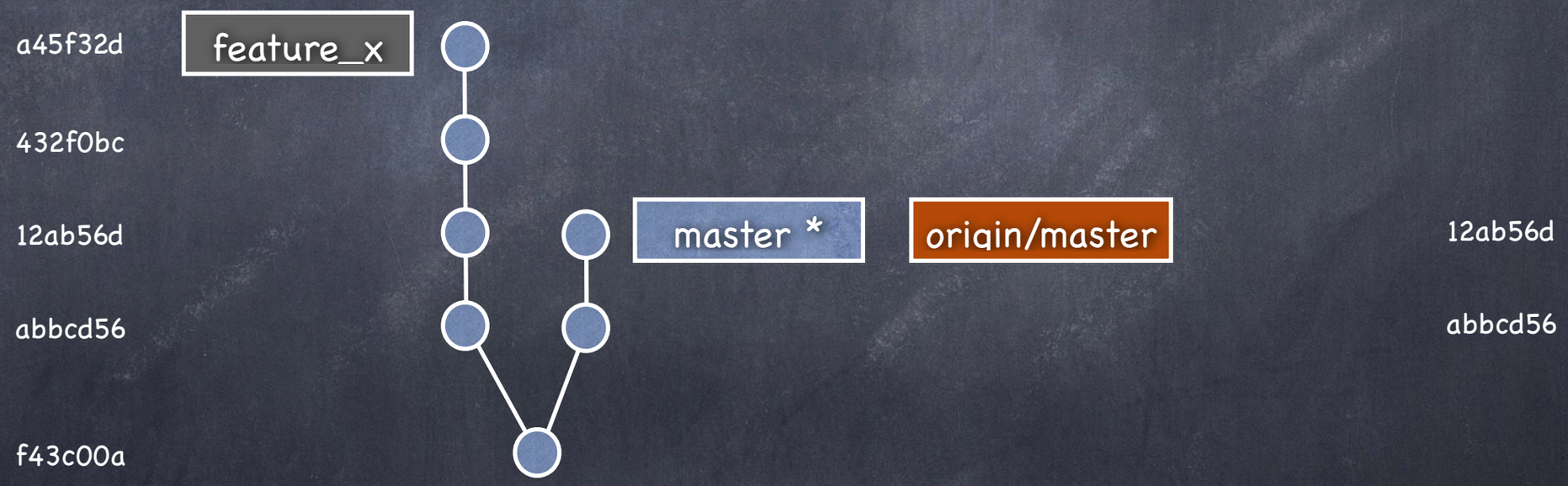
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

- ¿Qué pasa si mientras hacemos la integración en master producen nuevos cambios?

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge



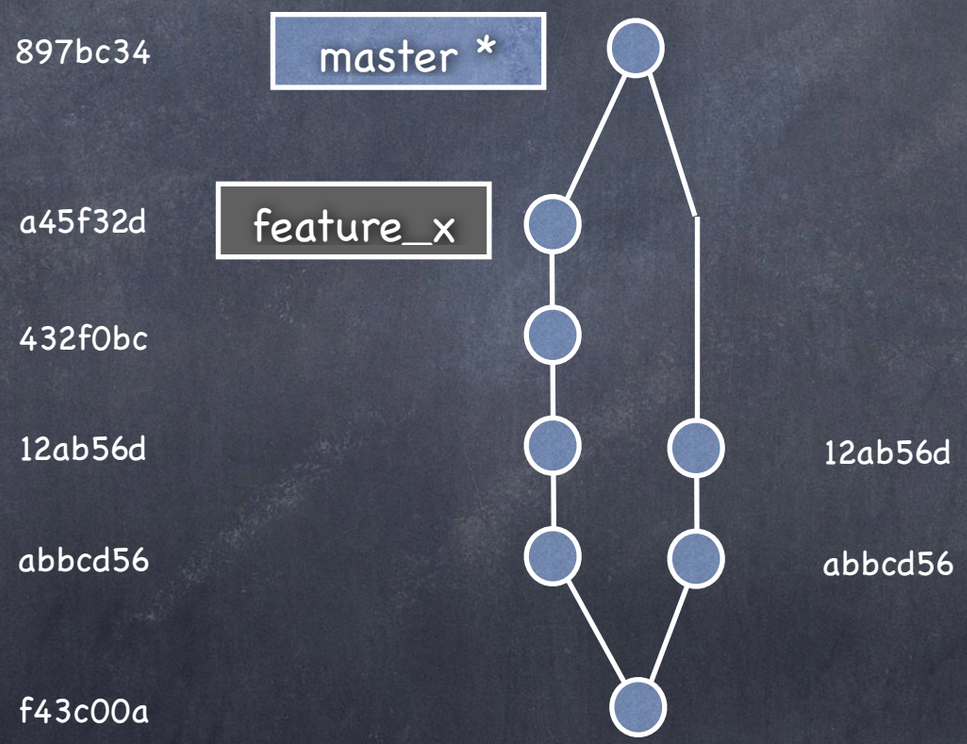
git
cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

git
aprendegit.com

merge

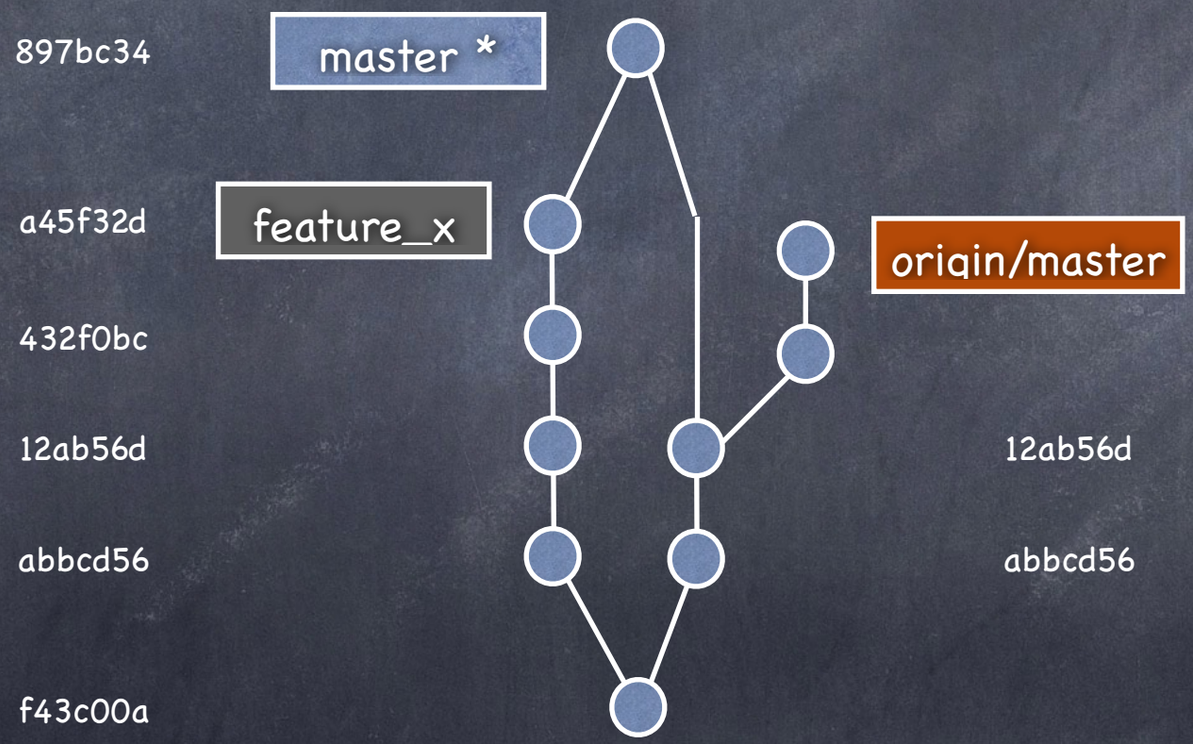
```
git checkout master
git merge feature_x
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

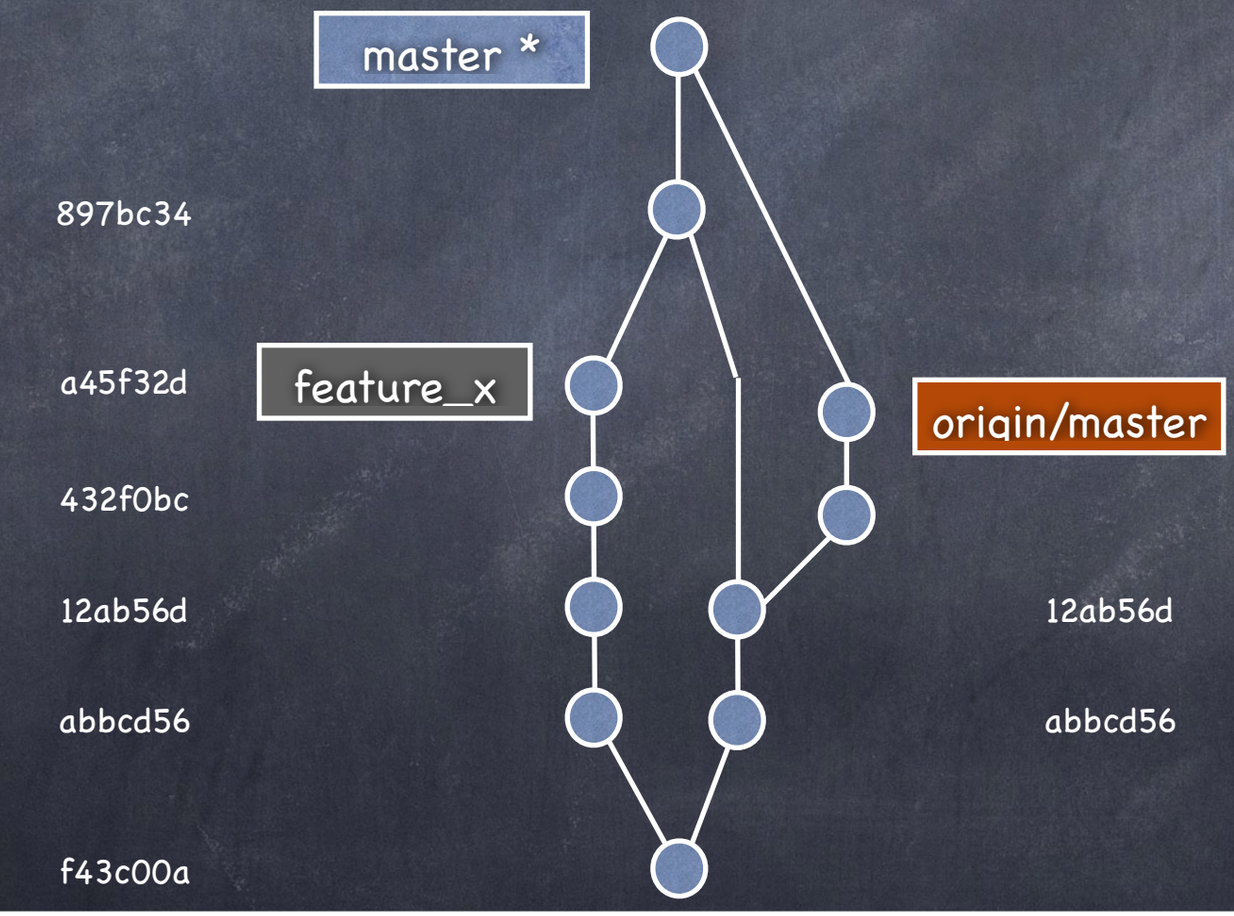
git fetch



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

merge

```
git checkout master
git merge origin/master
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

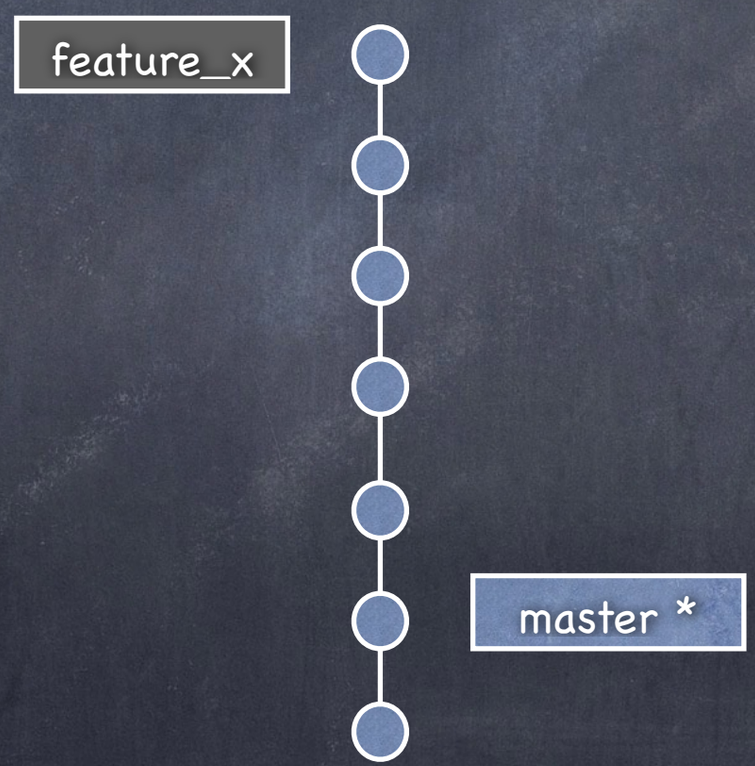
merge

- Documentación
- https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
- https://git-scm.com/docs/git-merge

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

fast forward

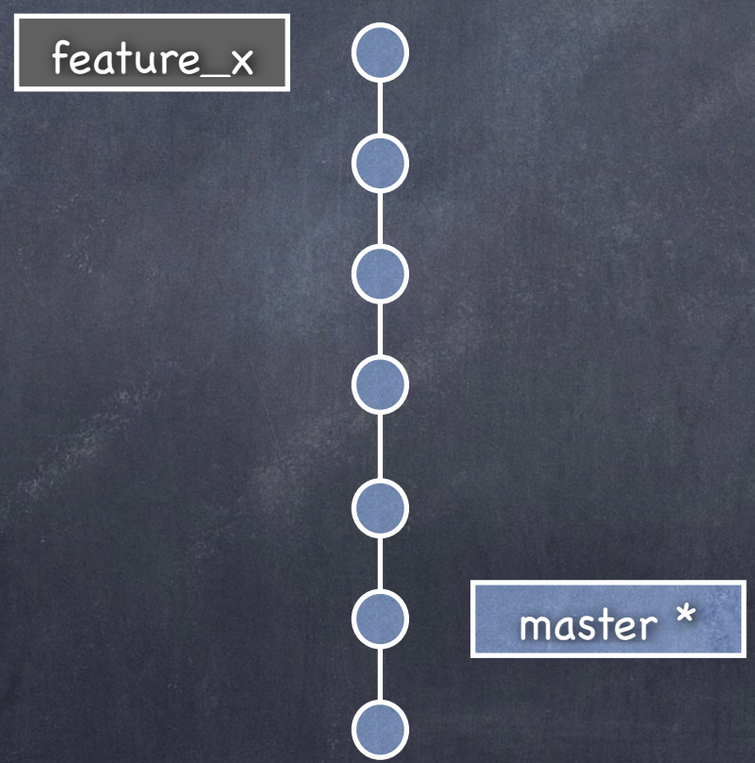
- Ocurre cuando podemos ir de una rama a otra siguiendo los padres de los commits



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

fast forward

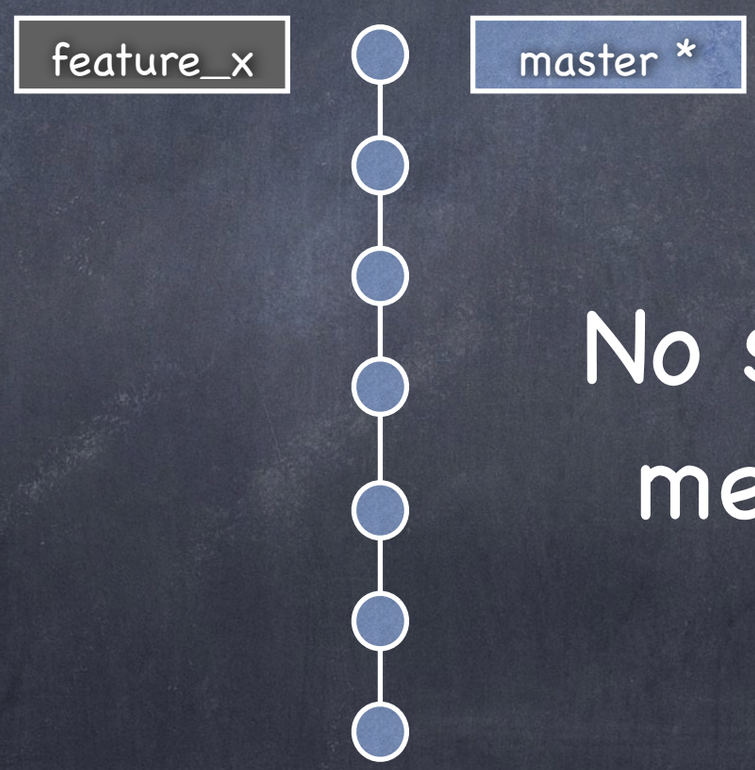
```
git checkout master
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una
Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

fast forward

```
git merge feature_x
```



No se genera un merge commit

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

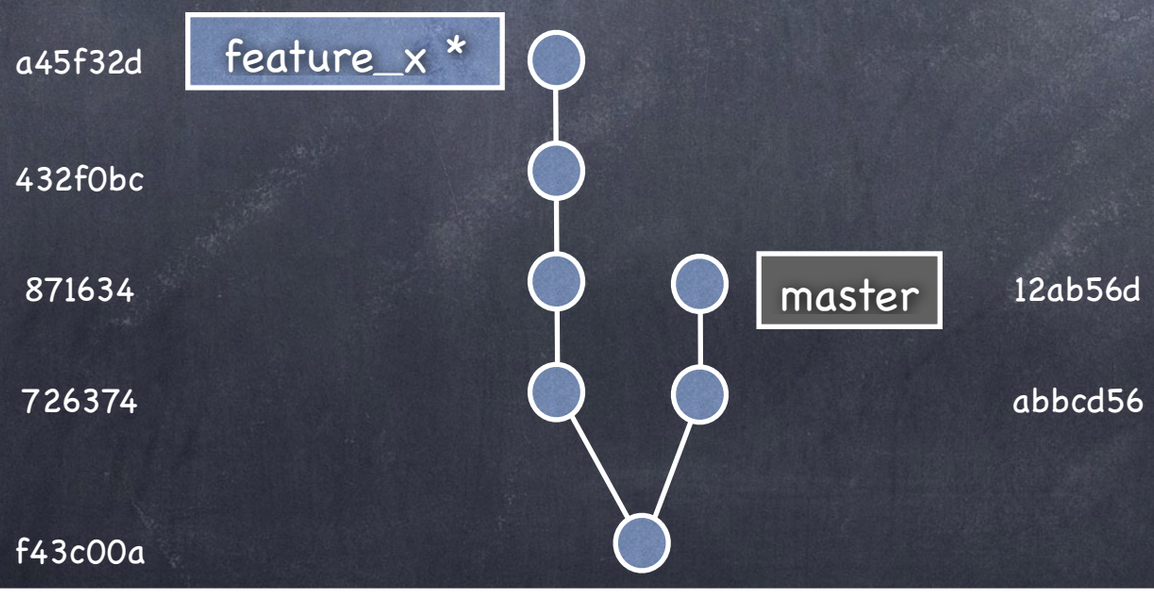
Rebase

- Operación que permite integrar cambios, al igual que merge
- De echo, se puede ver comuna alternativa al merge
- Para no equivocarse (cosa que es fácil):
  - "me voy a" la rama en la quiero integrar y "cojo" el código que quiero integrar

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Rebase

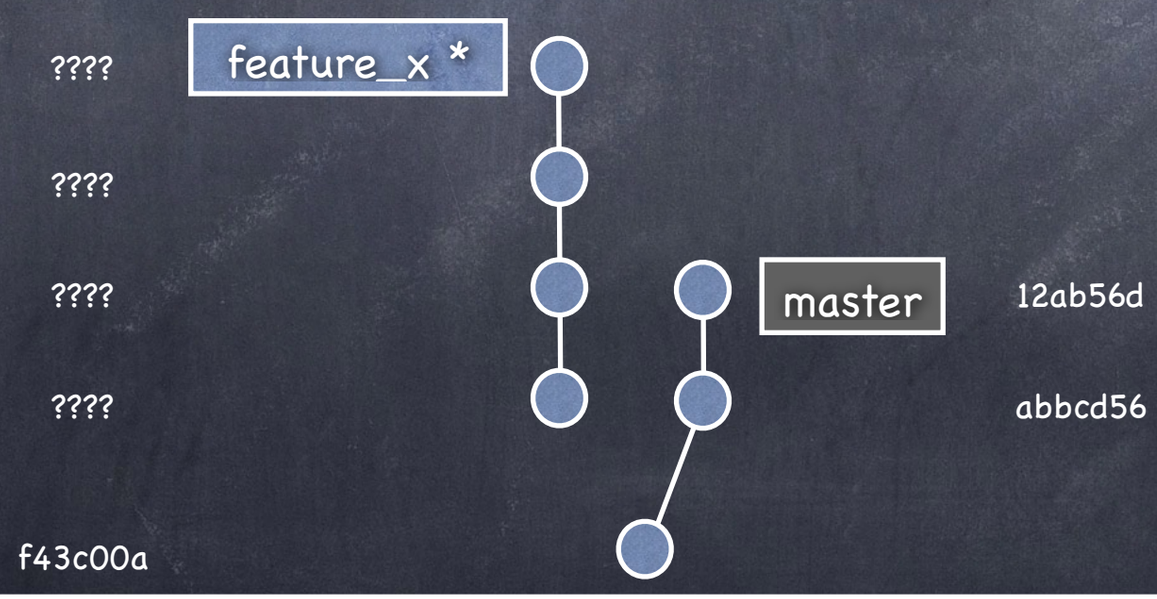
```
git checkout feature_x
  git rebase master
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

rebase

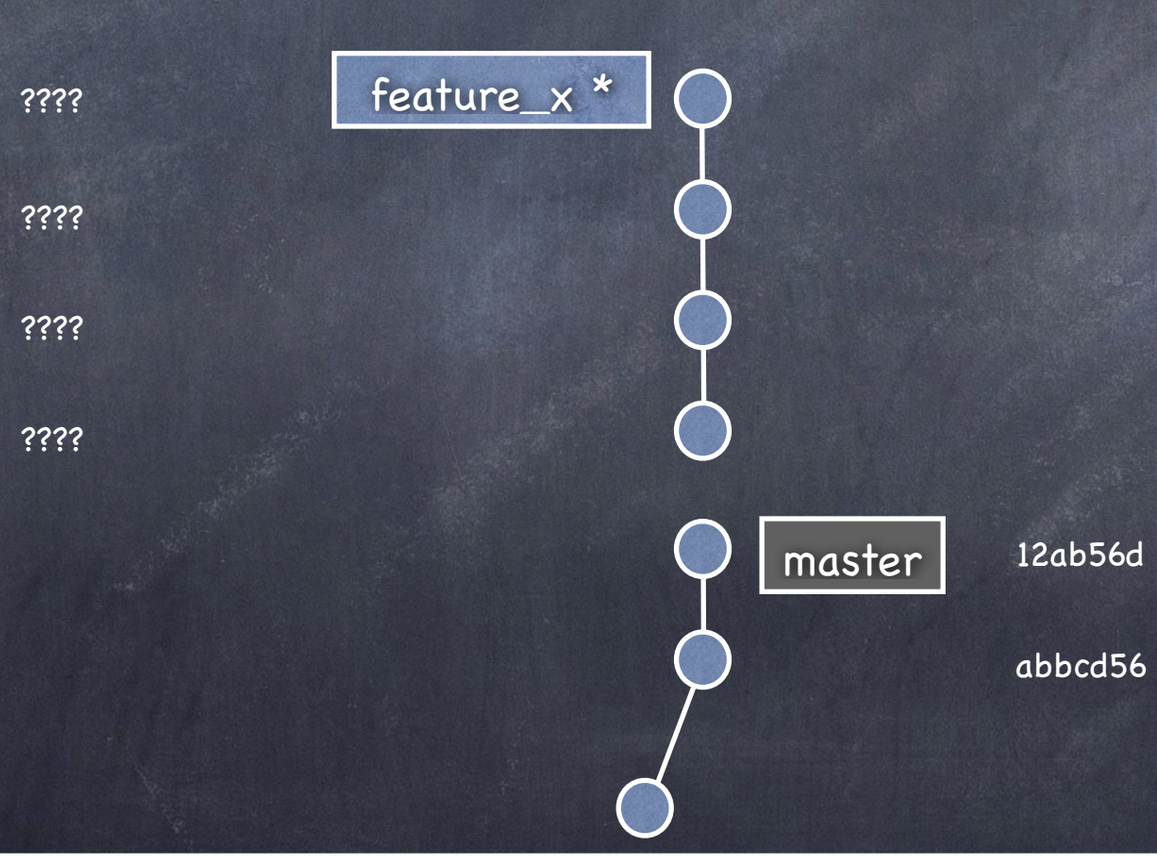
```
git checkout feature_x
  git rebase master
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

rebase

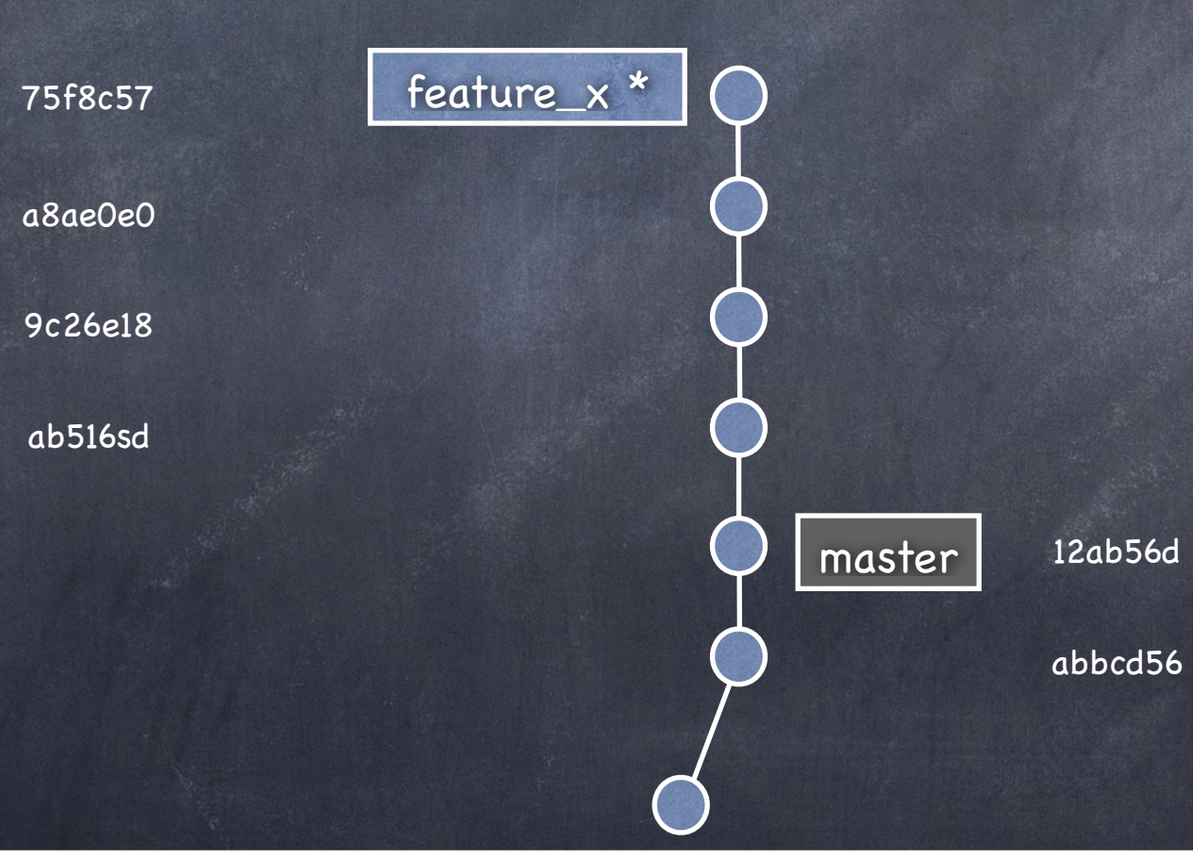
```
git checkout feature_x
  git rebase master
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

rebase

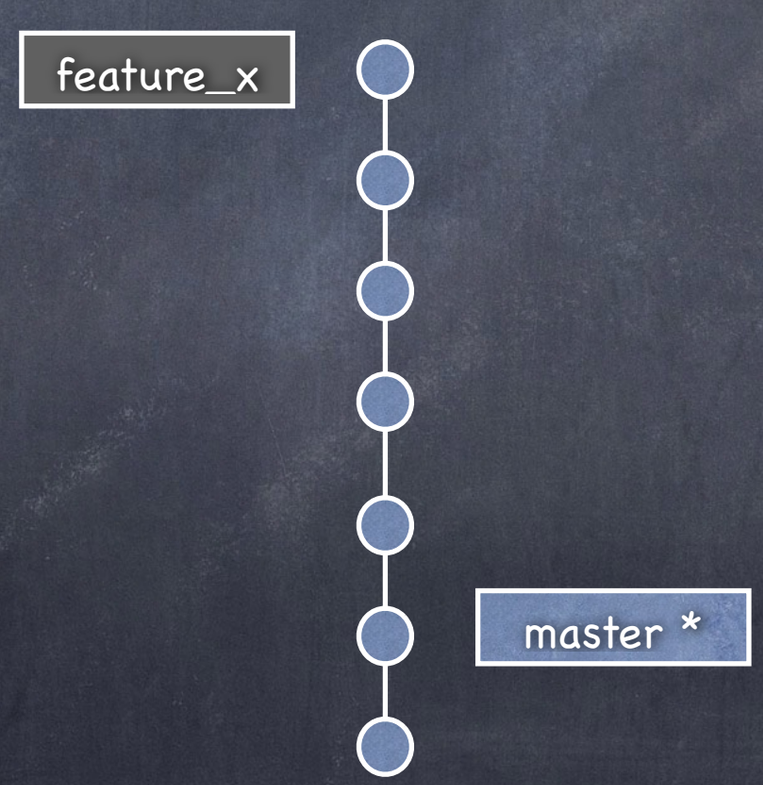
```
git checkout feature_x
  git rebase master
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

fast forward

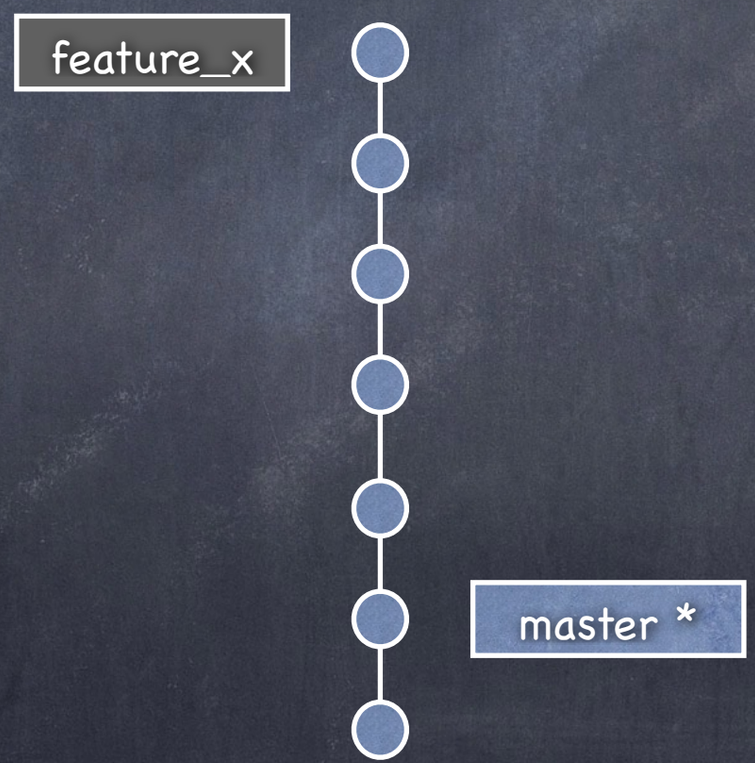
- Ocurre cuando podemos ir de una rama a otra siguiendo los padres de los commits



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

fast forward

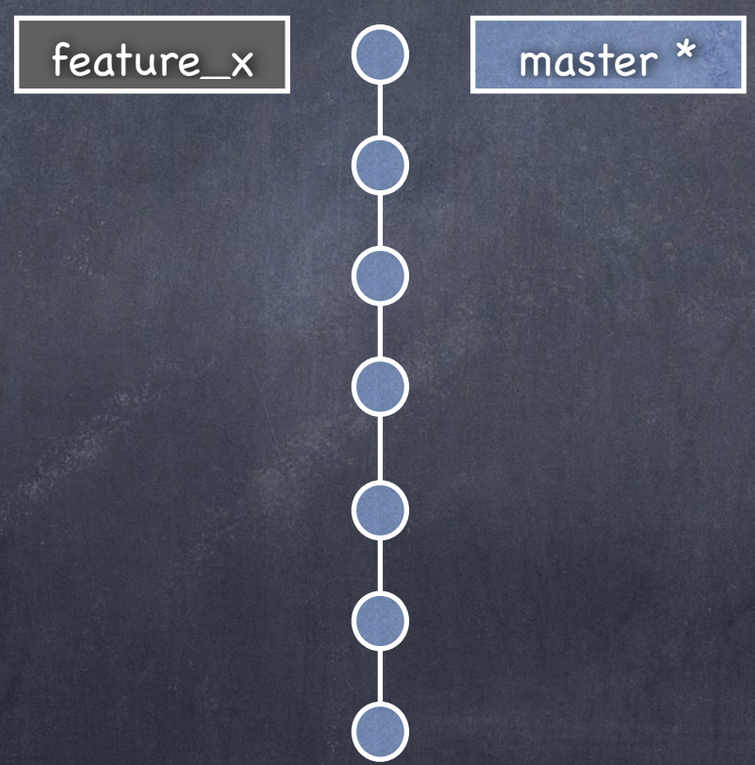
```
git checkout master
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una
Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

fast forward

```
git rebase feature_x
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

rebase

- Documentación
- https://git-scm.com/book/en/v2/Git-Branching-Rebasing
- https://git-scm.com/docs/git-rebase

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- Ocurren cuando, bien al mezclar dos snapshots (merge) o aplicar un parche (cherry-pick, stash, rebase, revert...) git no sabe cómo mezclar el texto de los ficheros

  (normalmente porque la misma zona de un fichero ha sido modificada simultáneamente)

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- Cuando se producen conflictos, git nos avisa de diferente manera, por ejemplo, si se produce un conflicto al hacer merge:

```
# git merge feature_1
Auto-merging config.ru
CONFLICT (content): Merge conflict in config.ru
Automatic merge failed; fix conflicts and then commit the result.
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- Para ver qué ficheros han resultado con conflicto usamos git-status

```
# git status
On branch feature_2
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   config.ru

no changes added to commit (use "git add" and/or "git commit -a")
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- Dentro de los ficheros con conflictos, git añade los marcadores estándar <<< === >>> para indicar qué zonas del fichero tienen conflictos

```
<<<<<<< HEAD
# This file is used by Rack-based servers to start the application.
# fasdf
require ::FileUtils.expand_path('../config/environment',  __FILE__)
run Blog::Application
========
require ::File.expand_path('../config/environment',  __FILE__)
run Test::Application
>>>>>>> feature_1
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- Para indicar que los conflictos de un fichero han sido resueltos, hay que añadir el fichero al staging area

En el ejemplo anterior: git add config.ru

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- Una vez se ha marcado el fichero como resuelto, añadiéndole al staging area, seguimos con la operación
- Si el conflicto ocurrió en un merge, basta con hacer un commit
- Si el conflicto ocurrió en un rebase, revert o cherry-pick, usamos la opción --continue de cada comando (git rebase —continue, etc)

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- Si la resolución de conflictos no se pude realizar en ese momento, por ejemplo por que requiere hablar con algún compañero del equipo, abortamos la operación:

```
git merge --abort
git rebase --abort
git revert --abort
git cherry-pick --abort
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- Si queremos utilizar una herramienta externa para comparar los ficheros, podemos ejecutar el comando

  git mergetool --tool=<herramienta>

- Por ejemplo, si has instalado meld en tu sistema:

  git mergetool --tool=meld

  Lanzaría meld para hacer la comparación de los ficheros

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- Por último, usando el comando checkout podemos resolver un conflicto descartando los cambios de uno u otro lado del merge:

nos quedamos con los cambios de HEAD
git checkout --ours config.ru

nos quedamos con los cambios de la rama que mergeamos
git checkout --theirs config.ru

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Resolución de conflictos

- https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging#Basic-Merge-Conflicts
- https://git-scm.com/book/en/v2/Git-Tools-Advanced-Merging#Merge-Conflicts
- https://git-scm.com/docs/git-mergetool

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es HEAD?

- HEAD es un puntero a la rama activa
- Eso significa que cuando creemos un nuevo commit, la rama a la que apunta HEAD pasará a apuntar al nuevo commit que acabamos de crear

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es HEAD?

- ¿Cómo podemos ver cuál es la rama activa?
- git branch nos muestra la rama activa en color verde o con un * delante

```
Alfonsos-MBP-2:repo aalba$ git branch
  features/PROJ-1234
  features/PROJ-5678
* master
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es HEAD?

- ¿Cómo podemos ver cuál es la rama activa?
  - El comando git status también nos indica cuál es la rama activa

```
Alfonsos-MBP-2:repo aalba$ git status
On branch master
nothing to commit, working directory clean
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es HEAD?

- ¿Cómo podemos ver cuál es la rama activa?
  - Mirando el contenido del fichero .git/HEAD

```
Alfonsos-MBP-2:repo aalba$ cat .git/HEAD
ref: refs/heads/master
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es eso de detached HEAD?

- Cuando un repositorio no tiene rama activa, decimos que se encuentra en un estado "detached HEAD"
- ¿Cuándo ocurre eso?
  - Cuando hacemos checkout de un commit al que no apunta directamente ninguna rama
  - Cuando hacemos checkout de un tag
  - Cuando hacemos checkout de una rama remota

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

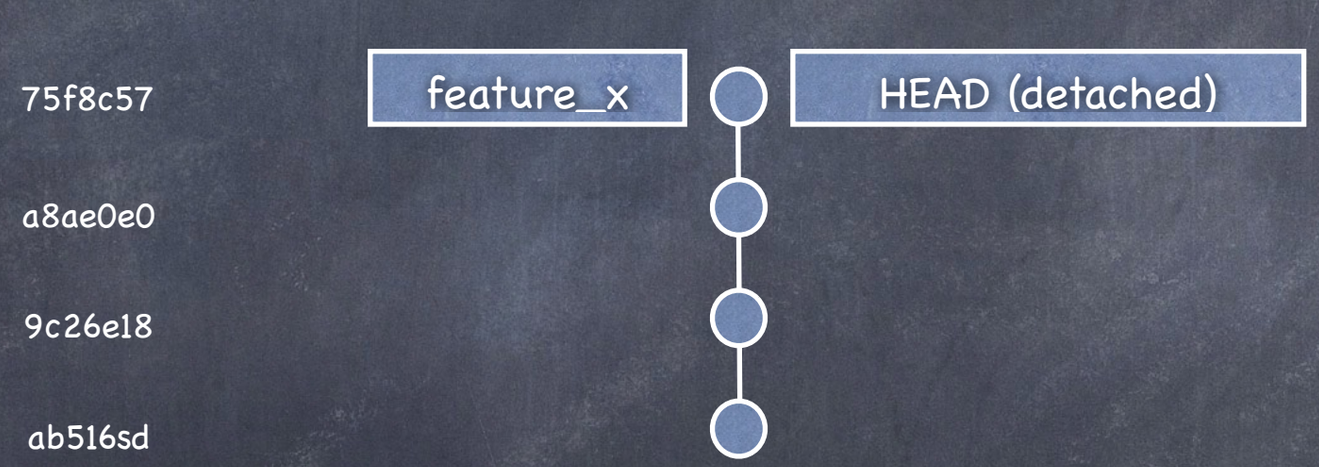
¿Qué es eso de detached HEAD?

- Si en estando en un estado "detached HEAD" hacemos un commit nuevo, este commit no quedará reverenciado por ninguna rama. Será HEAD directamente el puntero que apunte al nuevo commit.

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es eso de detached HEAD?

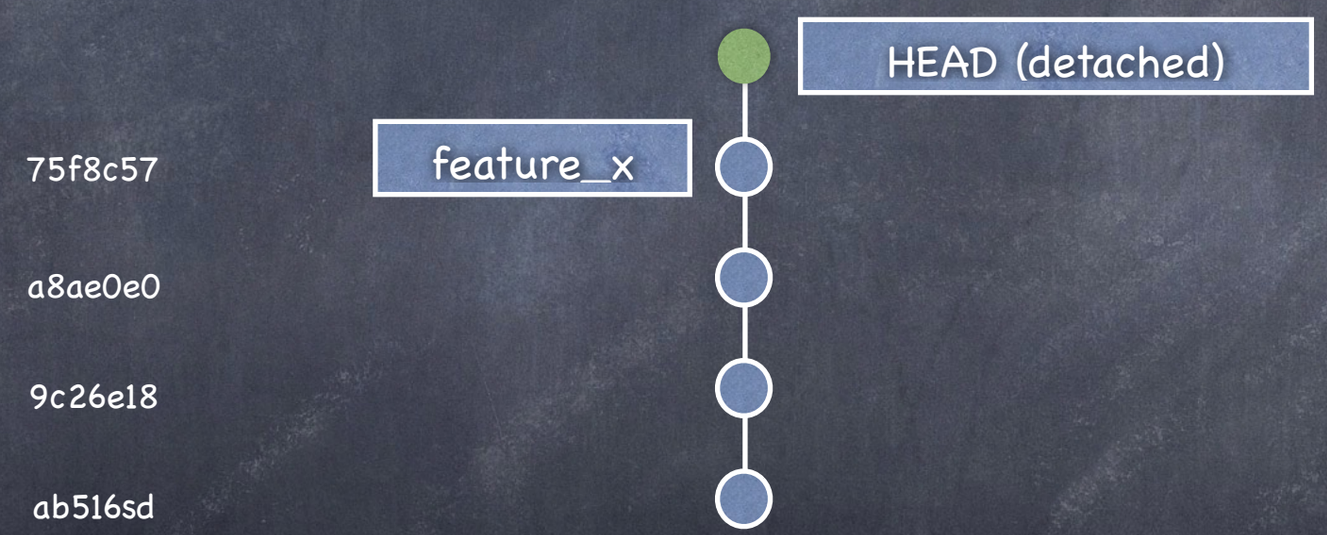
NO HAY RAMA ACTIVA



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es eso de detached HEAD?

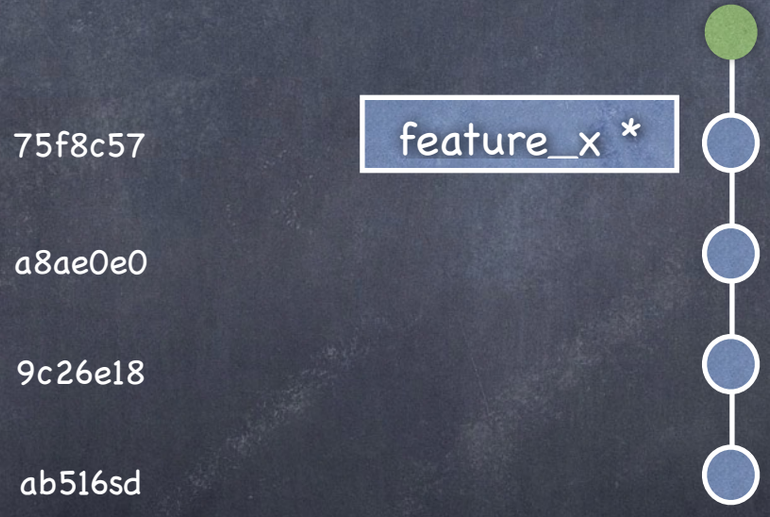
Al hacer commit, es HEAD el puntero que apunta al nuevo commit



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es eso de detached HEAD?

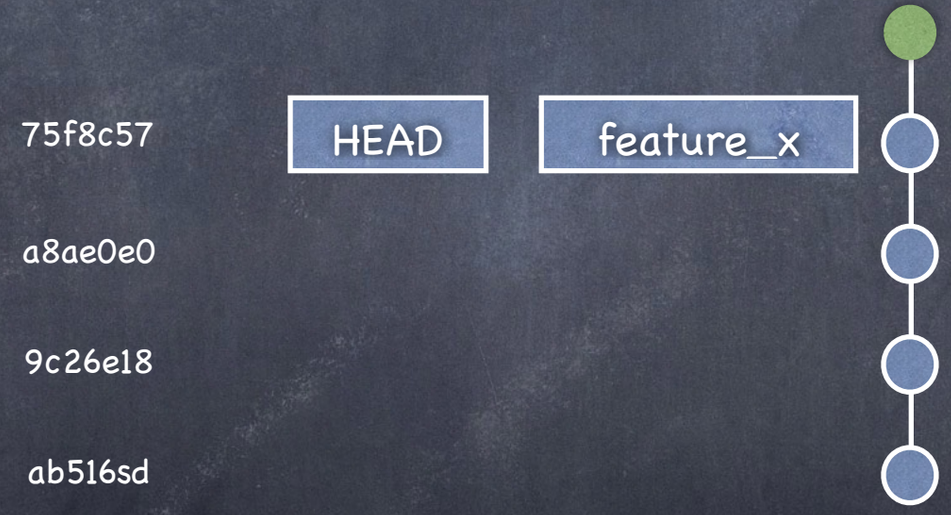
Si en esta situación hacemos checkout de la rama feature_x ...



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es eso de detached HEAD?

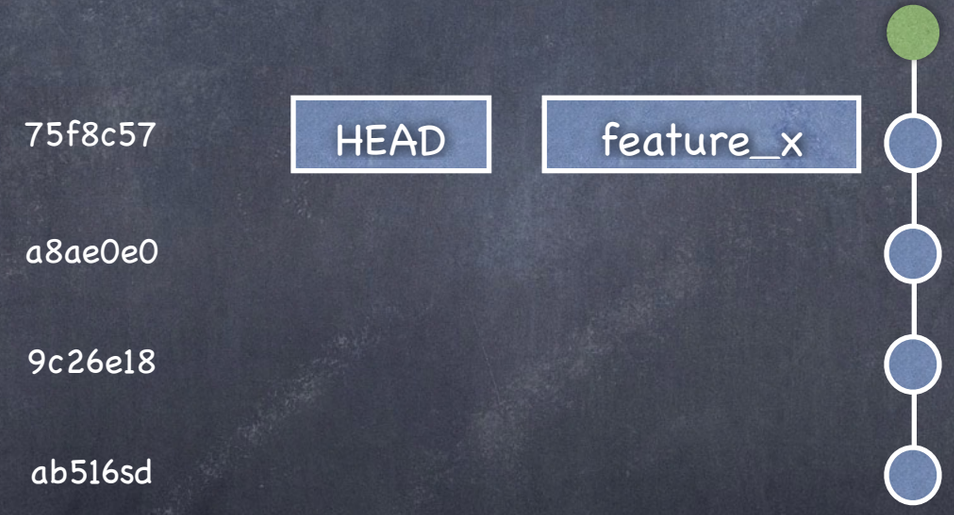
HEAD vuelve a apuntar a feature_x y dejamos de ver el último commit ya que no tenemos ningún puntero que lo referencie..



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es eso de detached HEAD?

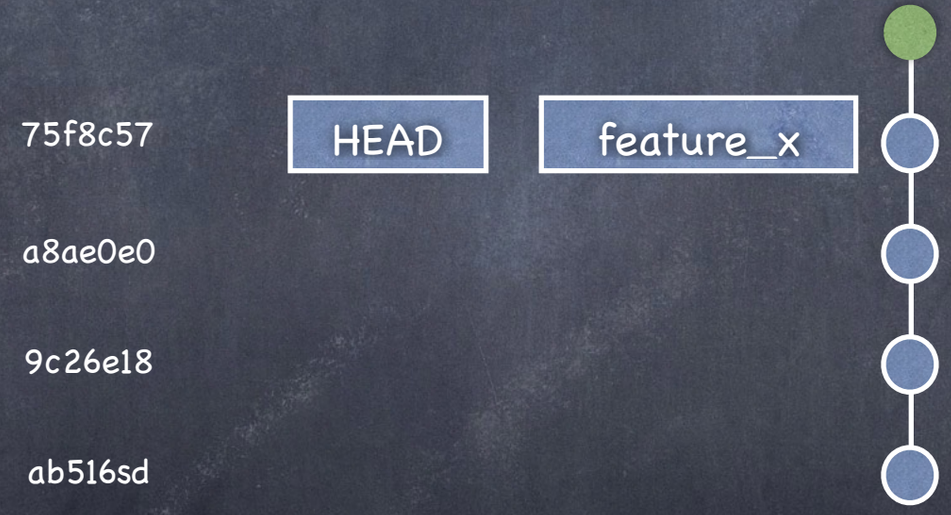
¿cómo recuperamos ese commit que acabamos de "perder"?



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es eso de detached HEAD?

Basta crear una rama o tag que lo referencie con el comando git-branch o git-tag



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es eso de detached HEAD?

Esto es lo que ocurre cuando pasamos a tener un detached HEAD

```
Alfonsos-MBP-2:repo aalba$ git checkout 44d15bb
Note: checking out '44d15bb'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 44d15bb... Initial commit
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¿Qué es eso de detached HEAD?

Si hemos hecho algún commit desde un estado "detached HEAD" y hacemos checkout,
git se da cuenta de que nos hemos dejado ese commit verde
¡y nos avisa!

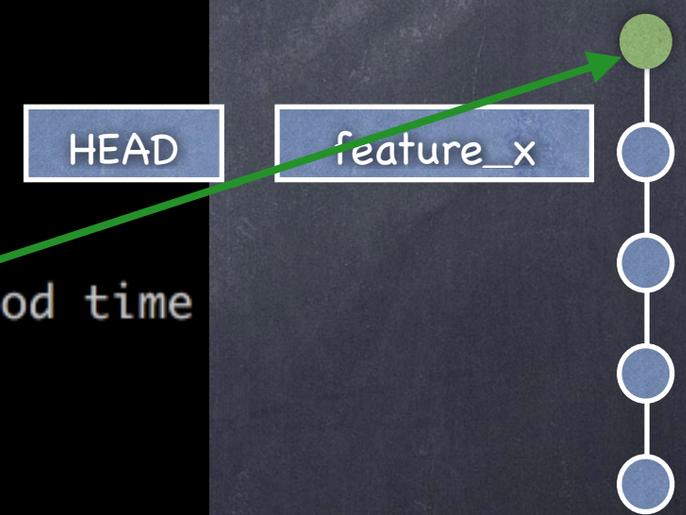
```
Alfonsos-MBP-2:repo aalba$ git checkout master
Warning: you are leaving 1 commit behind, not connected to
any of your branches:

  4d6c685 Nuevo sistema de mensajería.

If you want to keep it by creating a new branch, this may be a good time
to do so with:

 git branch <new-branch-name> 4d6c685

Switched to branch 'master'
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset

- El comando reset sirve para:
  - Sacar modificaciones del staging area
  - Mover ramas locales

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

sacando cambios del staging area

- Dados una serie de modificaciones que se han subido al staging area...

```
On branch integracion
Your branch is ahead of 'origin/integracion' by 1 commit.
  (use "git push" to publish your local commits)
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   index.html
        new file:   servicios.html
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

sacando cambios del staging area

git reset HEAD

```
On branch integracion
Your branch is ahead of 'origin/integracion' by 1 commit.
  (use "git push" to publish your local commits)
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

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

        servicios.html
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

sacando cambios del staging area

git reset HEAD

```
On branch integracion
Your branch is ahead of 'origin/integracion' by 1 commit.
  (use "git push" to publish your local commits)
Changes not staged for commit:

        modified:   index.html

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

        servicios.html
```

Sacamos todas las modificaciones del staging area

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

sacando cambios del staging area

git reset HEAD

```
On branch integracion
Your branch is ahead of 'origin/integracion' by 1 commit.
  (use "git push" to publish your local commits)
Changes not staged for commit:

        modified:   index.html

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

        servicios.html
```

Sacamos todas las modificaciones del staging area

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

sacando cambios del staging area

```
git reset HEAD [rutas a los ficheros]
```

```
On branch integracion
Your branch is ahead of 'origin/integracion' by 1 commit.
  (use "git push" to publish your local commits)
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   servicios.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

sacando cambios del staging area

```
git reset HEAD [rutas a los ficheros]
```

```
On branch integracion
  (use "git reset HEAD <file>..." to unstage)

        new file:   servicios.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html
```

Sacamos sólo el fichero index.html del staging area

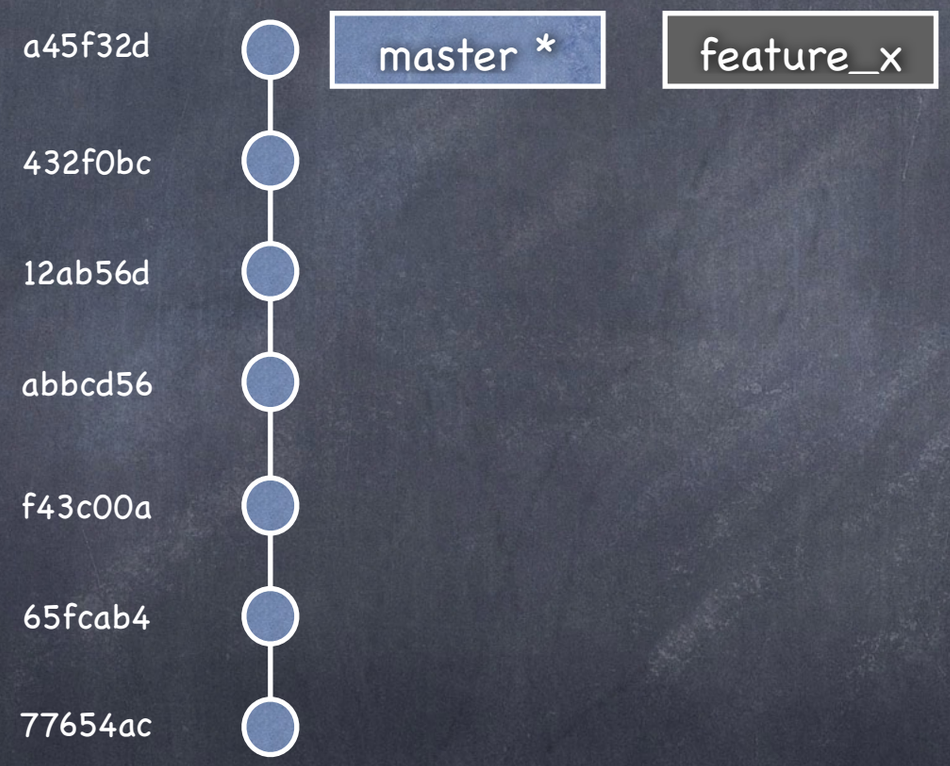
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset --hard

- Este comando mueve la rama activa al commit que se le indique
- Al mover la rama activa al commit de destino, limpia el área de trabajo y extrae el snapshot del commit de destino

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset --hard



cursodegit.com

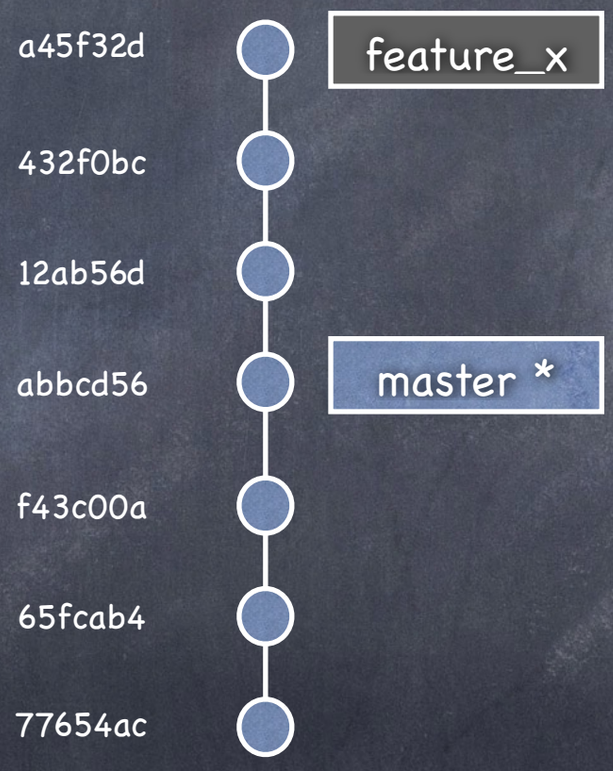
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

reset --hard

```
git reset --hard abbcd56
```

Al terminar la operación el, área de trabajo contendrá el snapshot del commit abbcd56



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset --hard

- La operación reset --hard es destructiva
- Para poder hacer la operación, el comando reset limpia el área de trabajo para sacar el snapshot
- Al hacer esta "limpieza" git borrará cualquier modificación existente en el área de trabajo
- Estas modificaciones no se pueden recuperar

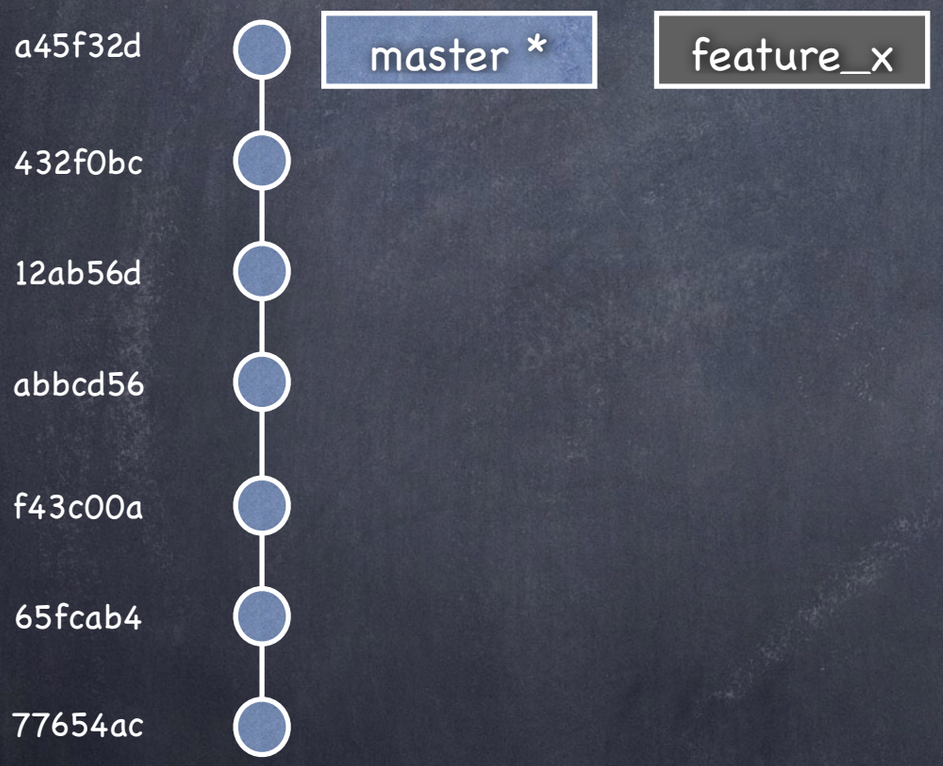
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset --soft

- Este comando mueve la rama activa al commit que se le indique
- Al mover la rama activa al commit de destino, "calcula" la diferencia entre el snapshot de destino y el de origen
- Deja esta diferencia en el área de trabajo

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset --soft



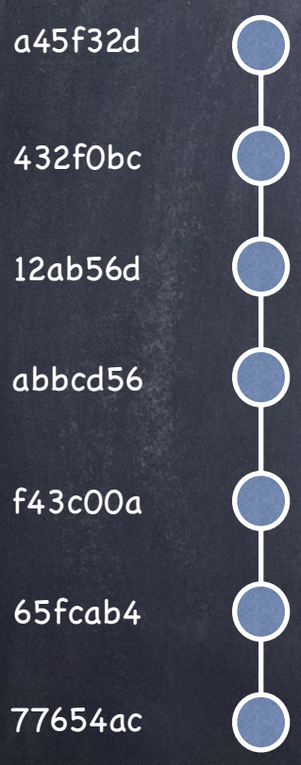
cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

reset --soft

```
git reset --soft abbcd56
```




```
On branch integracion
Your branch is ahead of 'origin/integracion' by 2 commits.
  (use "git push" to publish your local commits)
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   contacto.html
        modified:   servicios.html

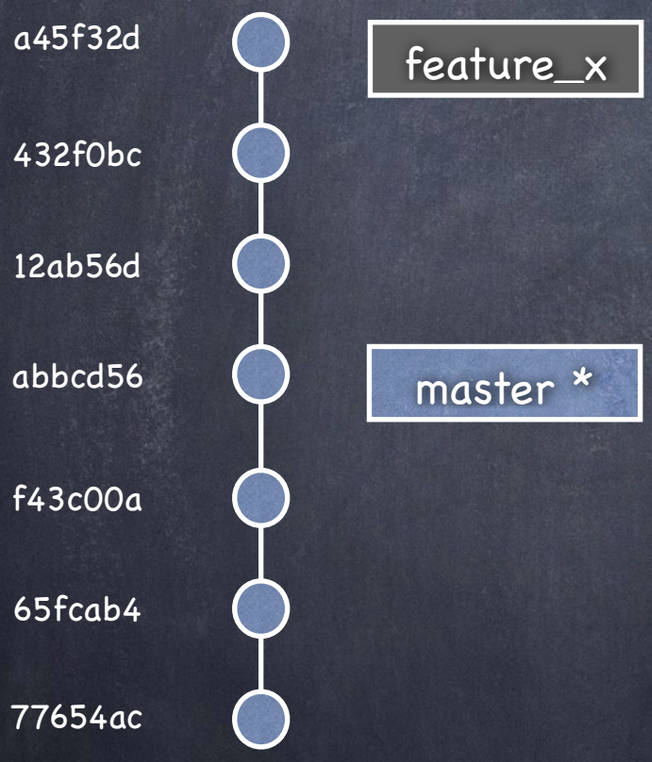
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        quienes_somos.html
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset --soft

```
git reset --soft abbcd56
```



Los cambios quedan en el área de trabajo

```
On
You
  (
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   contacto.html
        modified:   servicios.html

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

        quienes_somos.html
```

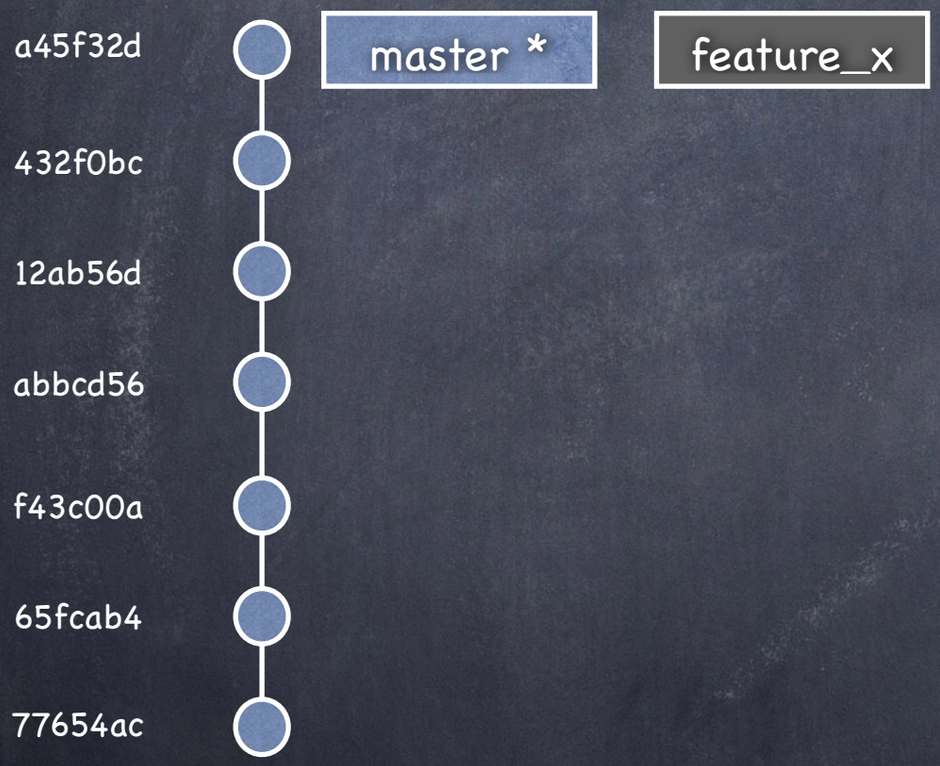

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset --mixed

- Este comando mueve la rama activa al commit que se le indique
- Al mover la rama activa al commit de destino, "calcula" la diferencia entre el snapshot de destino y el de origen
- Deja esta diferencia en el staging area

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset --mixed



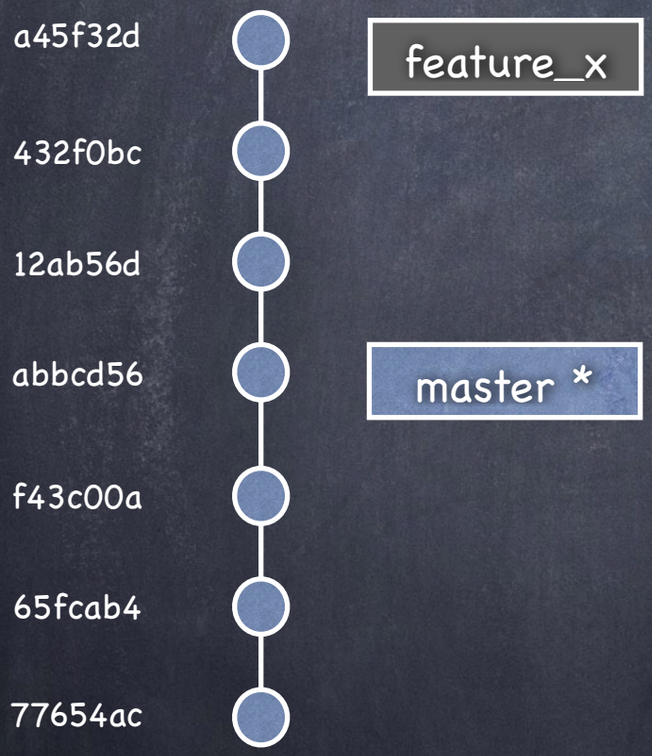
cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

reset --mixed

```
git reset --mixed abbcd56
```



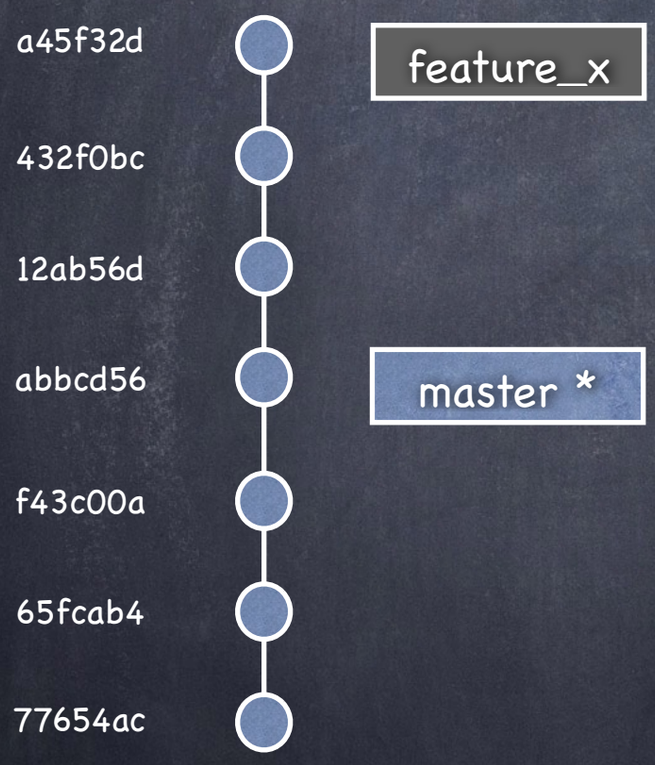

```
On branch integracion
Your branch is ahead of 'origin/integracion' by 2 commits.
  (use "git push" to publish your local commits)
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   contacto.html
        new file:   quienes_somos.html
        modified:   servicios.html
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset --mixed

```
git reset --mixed abbcd56
```



Los cambios quedan en el staging area

```
On branch integracion
Your branch is ahead of 'origin/integracion' by 2 commits.
  (use "git push" to publish your local commits)
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   contacto.html
        new file:   quienes_somos.html
        modified:   servicios.html
```

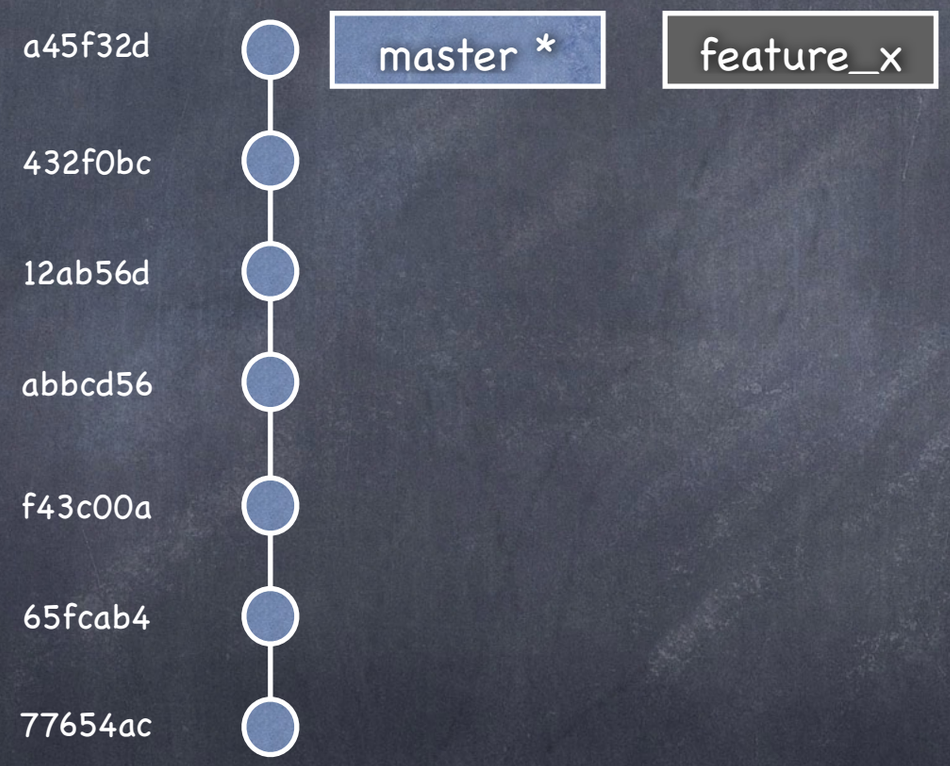

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

"perdiendo" commits

- La operación al hacer la operación reset, puede parecer que "perdemos" commits
- Recuerda que git no borra commit, lo que pasa es que no podemos verlos, porque hemos perdido las referencias que nos permitían llegar hasta ellos

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

"perdiendo" commits



cursodegit.com

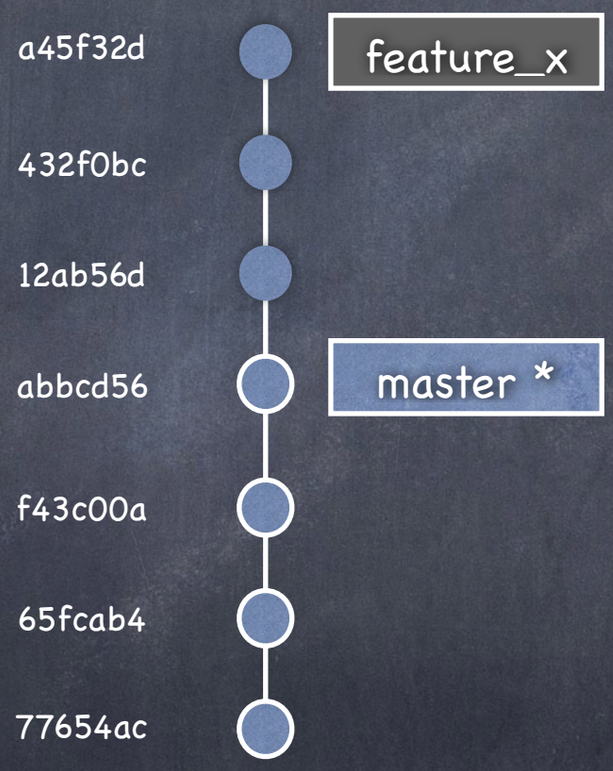
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

"perdiendo" commits

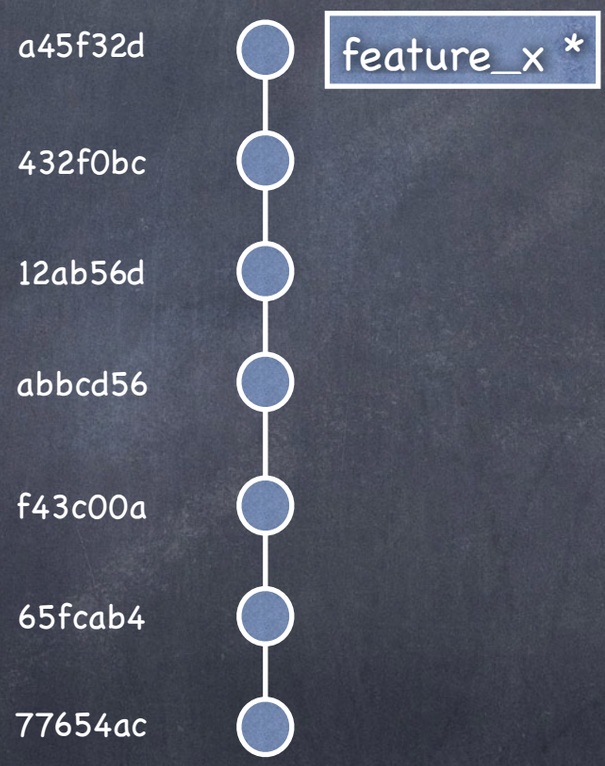
Vemos todos los commits porque la rama feature_x nos permite llegar a ellos

¿qué ocurriría si la rama feature_x no estuviese?



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

"perdiendo" commits



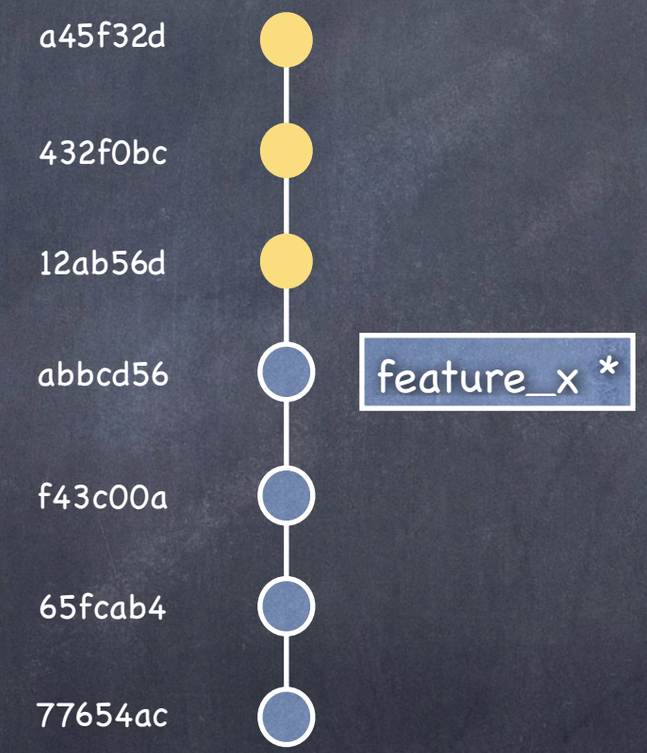
cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

"perdiendo" commits

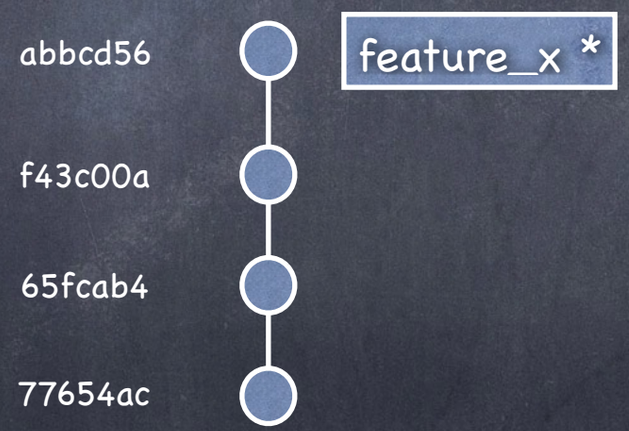
¡Dejamos de ver estos commits porque no tenemos ninguna referencia que nos permita llegar a ellos!



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

"perdiendo" commits

Parece que los hemos perdido, pero están ahí... es sólo que no podemos verlos



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

reset

- Reset demystified: https://git-scm.com/blog/2011/07/11/reset.html
- https://git-scm.com/docs/git-reset

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Stashing

Stash: área en el que podemos guardar cambios en ficheros para aplicarlos luego

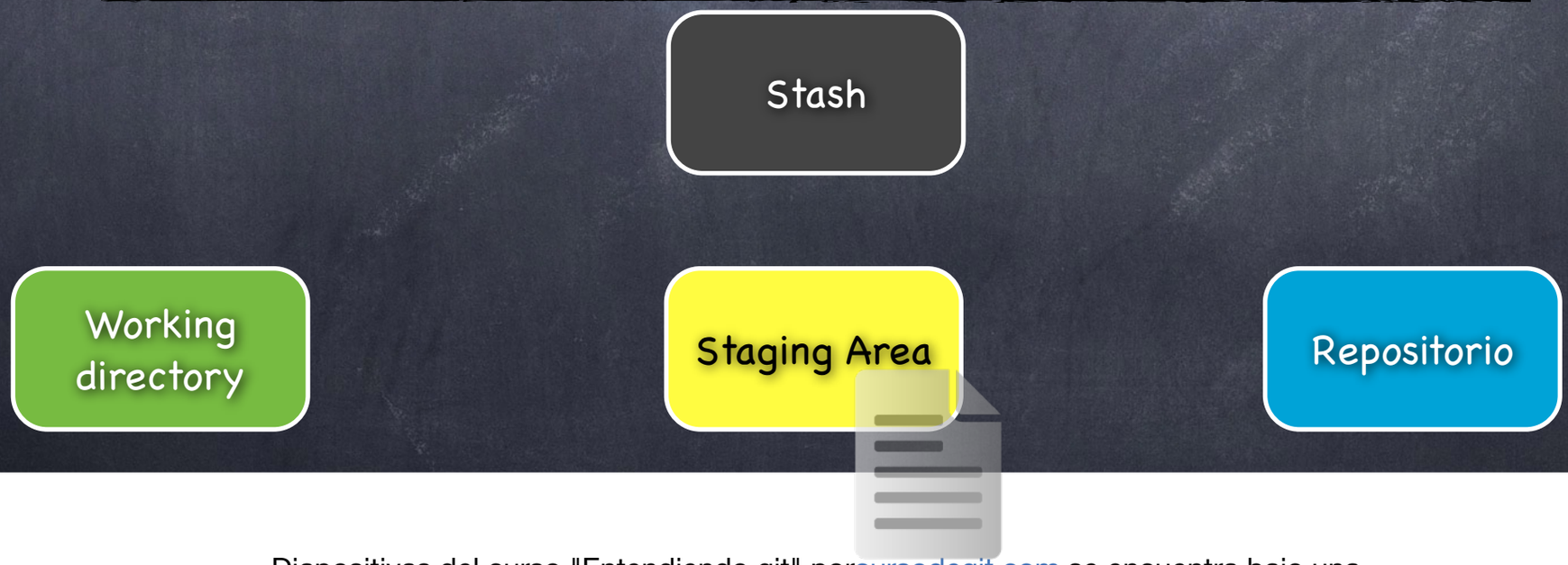
Útil para:
Poder interrumpir una tarea y trabajar en otra
Mover cambios de un sitio a otro

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Stashing

Almacenar en el stash los cambios del working copy en ese momento

```
git stash save “mensaje”
```



git
cursodegit.com

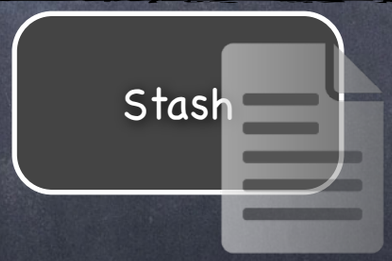
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

git
aprendegit.com

Stashing

Almacenar en el stash los cambios del working copy en ese momento

```
git stash save “mensaje”
```



Working directory

Staging Area

Repositorio

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Stashing

Muestra el listado de entradas del stash

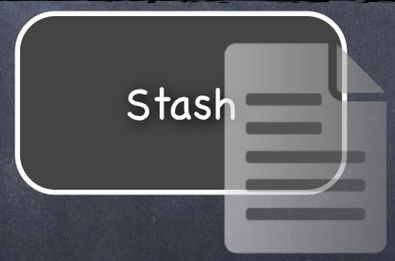
```
git stash list
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Stashing

Recuperar algo almacenado en el Stash

```
git stash pop <nombre>
```



Working directory

Staging Area

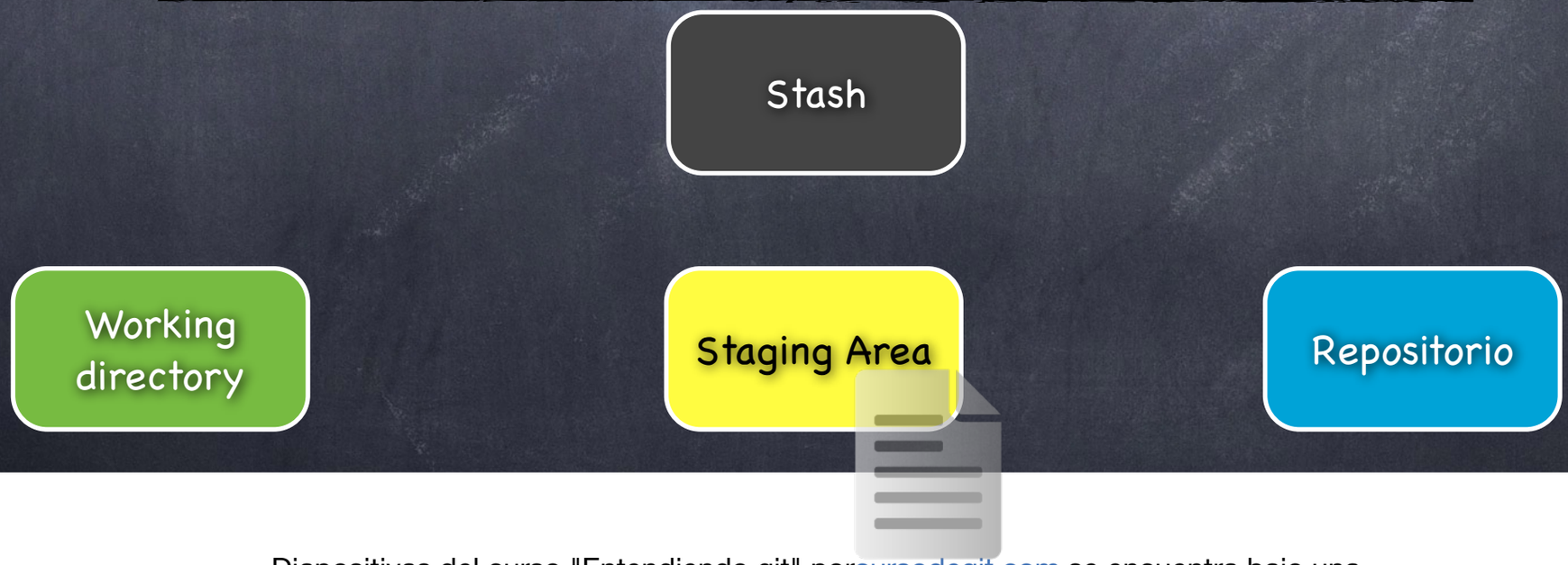
Repositorio

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Stashing

Recuperar algo almacenado en el Stash

```
git stash pop <nombre>
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Stashing

Recuperar una entrada del stash y guarda una copia

```
git stash apply <nombre>
```

Borrar una entrada del stash sin aplicarla

```
git stash drop <nombre>
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Stashing

- https://git-scm.com/docs/git-stash
- https://git-scm.com/book/es/v2/Git-Tools-Stashing-and-Cleaning

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando git-diff

- ¿para qué serviría git si no pudiésemos comparar ficheros entre diferentes commits o incluso commits completos entre sí?
- Para eso tenemos precisamente el comando git-diff

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando git-diff

- Comparar el área de trabajo con respecto a la versión que hay en el staging area

  git diff

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

```
git diff
diff --git a/Prueba.txt b/Prueba.txt
index 036589c..bf5a060 100644
--- a/Prueba.txt
+++ b/Prueba.txt
@@ -3,8 +3,3 @@ Nueva línea al cambio
 Nueva lína de prueba para hace el último commit y completar el tutorial,
 Se añade linea de tecxto
 otra linea
-
-31-10-2014 11:33 [operez]  Añado linea para el primer commit
-04-11-2014 19:19 [operez]  Añado linea para el segundo commit y completar
-09-11-2015 19:16 [carsava] Añado esta linea
diff --git a/config.ru b/config.ru
index 6ce27fe..782c392 100644
--- a/config.ru
+++ b/config.ru
@@ -1,4 +1,2 @@
-# This file is used by Rack-based servers to start the application.
-# fasdf
 require ::File.expand_path('../config/environment',  __FILE__)
-run Blog::Application
+run Test::Application
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando git-diff

- Comparar uno o varios ficheros del área de trabajo con respecto a la versión que hay en el staging area

git diff Prueba.txt

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando git-diff

- Comparar el staging area con otro commit:

git diff --cached <referencia> <rutas>

Si se omite la <referencia>, se usa el commit al que apunta HEAD y si se omiten las rutas se compara el snapshot completo

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando git-diff

- Compara el área de trabajo con otro commit

git diff <referencia> <rutas>

Por ejemplo: git diff 124acb09 Prueba.txt

Si se omiten las rutas, se compara el área de trabajo con el snapshot completo

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

El comando git-diff

- Para comparar dos commits

git diff <referencia> <referencia> <rutas>

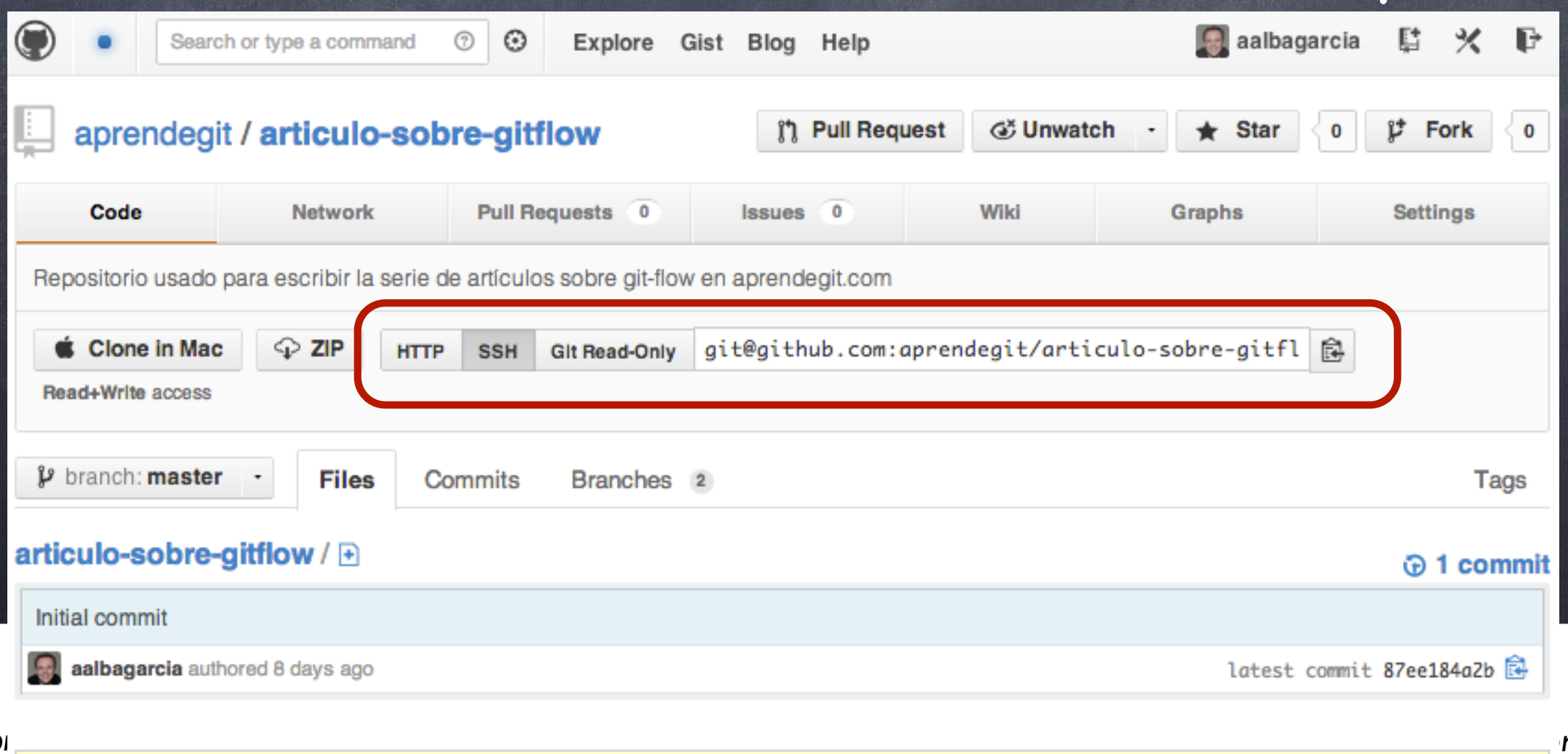
Ejemplos: git diff 16aefd0 feature_234

Si se omiten las rutas, se comparan los snapshots de código completos

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Repositorios remotos

Para subir nuestro repositorio a un repositorio remoto necesitamos conocer la URL del repositorio



git
cursodegit.co
git
rendegit.com

Repositorios remotos

Para añadir un repositorio remoto ejecutamos

```
git remote add <nombre> <url>
```

Se puede ver la lista de remotos con

```
git remote -v
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una
Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Repositorios remotos

Borrar un repositorio remoto

```
git remote remove <nombre>
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Push

Para subir todos nuestros commits de la rama activa al repositorio remoto ejecutamos

```
git push <remoto> <rama remota>
```



usando el repositorio que hemos creado durante la clase

```
git push origin master
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Tracking branches

Una "Tracking branch" es una rama local que está relacionada con una remota.

Si estás en una rama local que es "tracking branch" de una rama remota y ejecutas el comando

```
git push
```



git automáticamente lo convierte en

```
git push <tracking branch> <origin/rama remota>
```

Tracking branches

Podemos crear una "tracking branch" de una rama remota ejecutando

```
git branch --set-upstream <rama local> <remoto>/<rama remota>
```

que siguiendo con el ejemplo que hacemos en clase sería

```
git branch --set-upstream master origin/master
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Tracking branches

En cualquier momento podemos hacer que una tracking branch deje de serlo ejecutando

```
git branch --unset-upstream <rama>
```



En cualquier momento podemos hacer que una rama local sea una tracking branch ejecutando

```
git branch --set-upstream-to=<remoto>/<rama remota> <rama local>
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Pull / Fetch

Para descargarnos los commits del repositorio remoto

```
git fetch <remoto>
```

Este comando actualiza los commits únicamente. Si queremos ahora incorporar los cambios de una rama remota en una rama local ejecutamos

```
git checkout <rama local>
git merge <remoto>/<rama remota>
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Pull / Fetch

Supongamos que tenemos una rama local "feature_x" que es tracking branch de una rama remota "origin/feature_x"

La rama remota ha sido actualizada por otro miembro del equipo y queremos actualizarnos

```
git checkout feature_x
   git fetch origin
git merge origin/feature_x
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Pull / Fetch

Esta operación es tan común que git nos da un atajo

```
git checkout feature_x
git fetch origin
git merge origin/feature_x
```

```
git checkout feature_x
git pull
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Pull / Fetch

git-pull por defecto hace un merge, ¿quieres que en lugar de merge haga un rebase?

```
git checkout feature_x
git pull --rebase
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

pull-request

- Operación que consiste en solicitar a una persona, propietaria del repositorio A, que se conecte a nuestro repositorio (repositorio B) y incorpore en A los commits que están en B

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

pull-request

- ¿Cómo podemos realizar un pull request?
  - Usando las interfaces web de servicios como github, bitbucket, gitlab, etc
  - De forma manual añadiendo múltiples repositorios remotos y haciendo merge de las ramas remotas que queramos incorporar.

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

pull-request

- http://aprendegit.com/que-es-un-pull-request/

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras

- git-cherry-pick
- git-blame
- git-revert
- git-filter-branch
- Dropbox

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-cherry-pick

- Toma uno o más commits y aplica los cambios de cada commit en la rama activa
- Al hacerlo, crea un nuevo commit

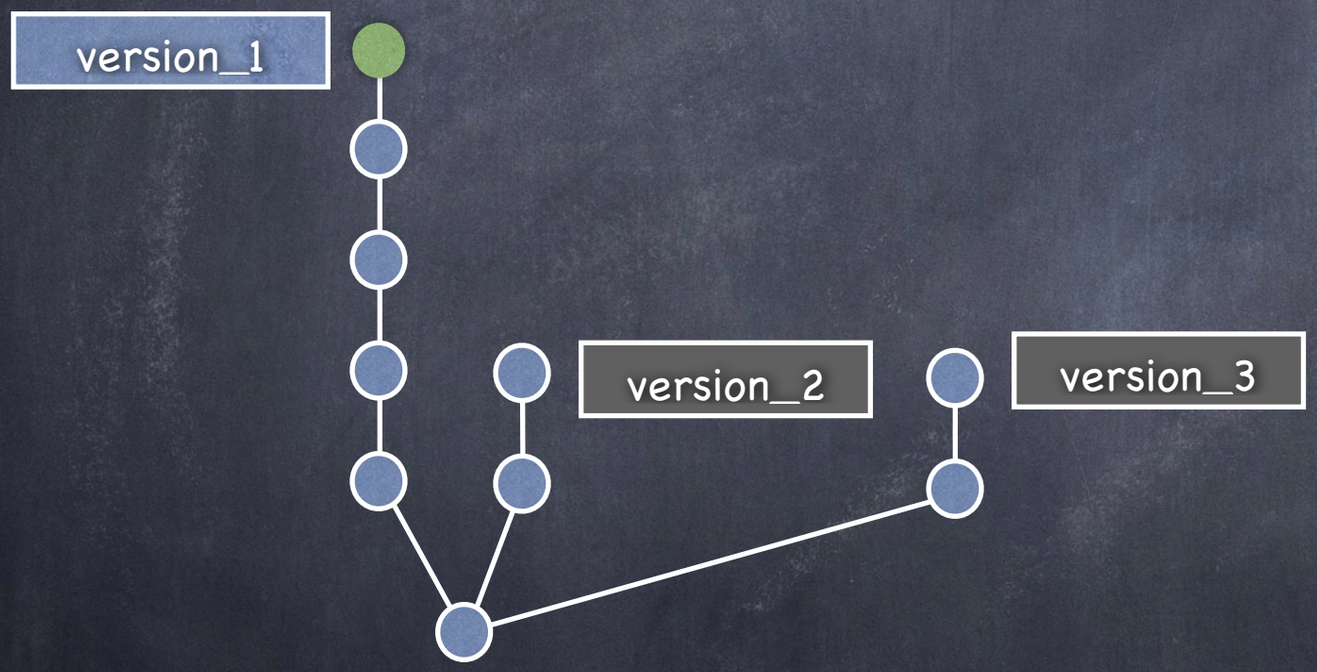
Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-cherry-pick

- Imagina que tienes un proyecto con tres versiones: 1, 2 y 3.
- Las versiones 1 y 2 están en mantenimiento, mientras la versión 3 es la que está en desarrollo activo
- Encuentras un bug en la versión 1 y lo corriges
- Ese mismo bug está presente en las versiones 2 y 3
- ¡No puedes hacer merge de la versión 2 o 3 con la versión 1!
- ¿Cómo te llevas el bug a la versión 2 o a la 3? ¡con un cherry-pick!

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-cherry-pick



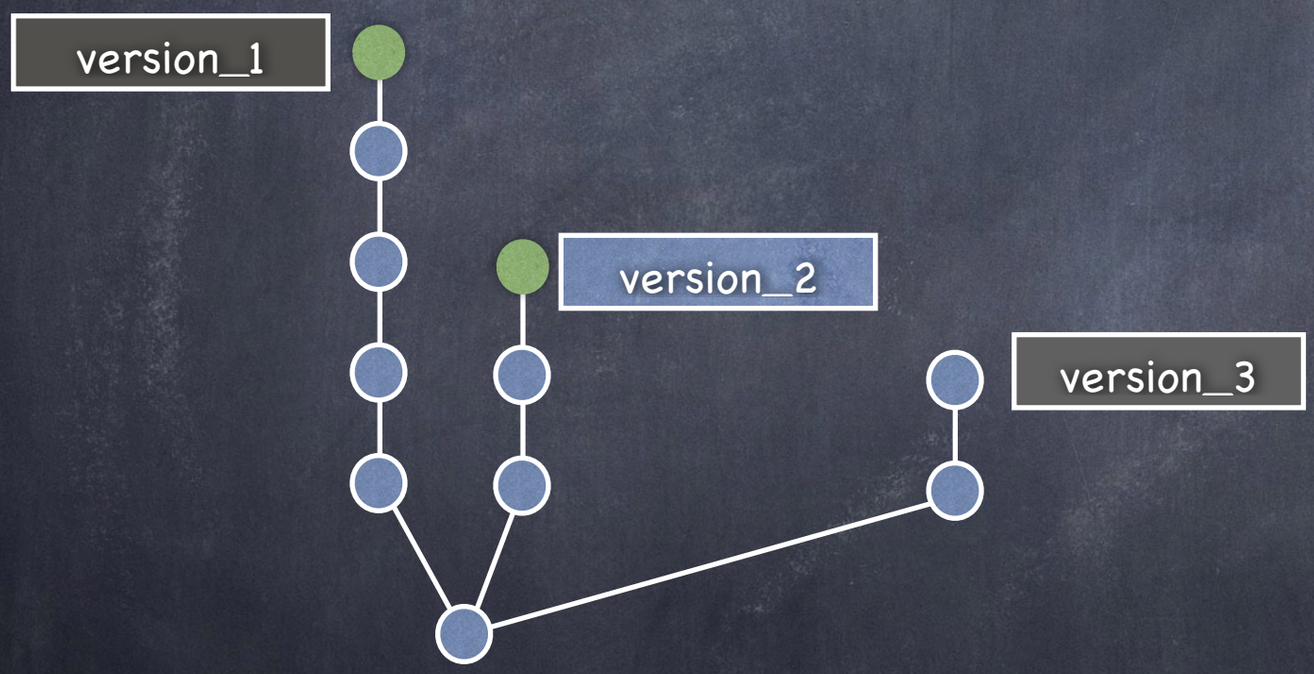
cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com

Extras: git-cherry-pick

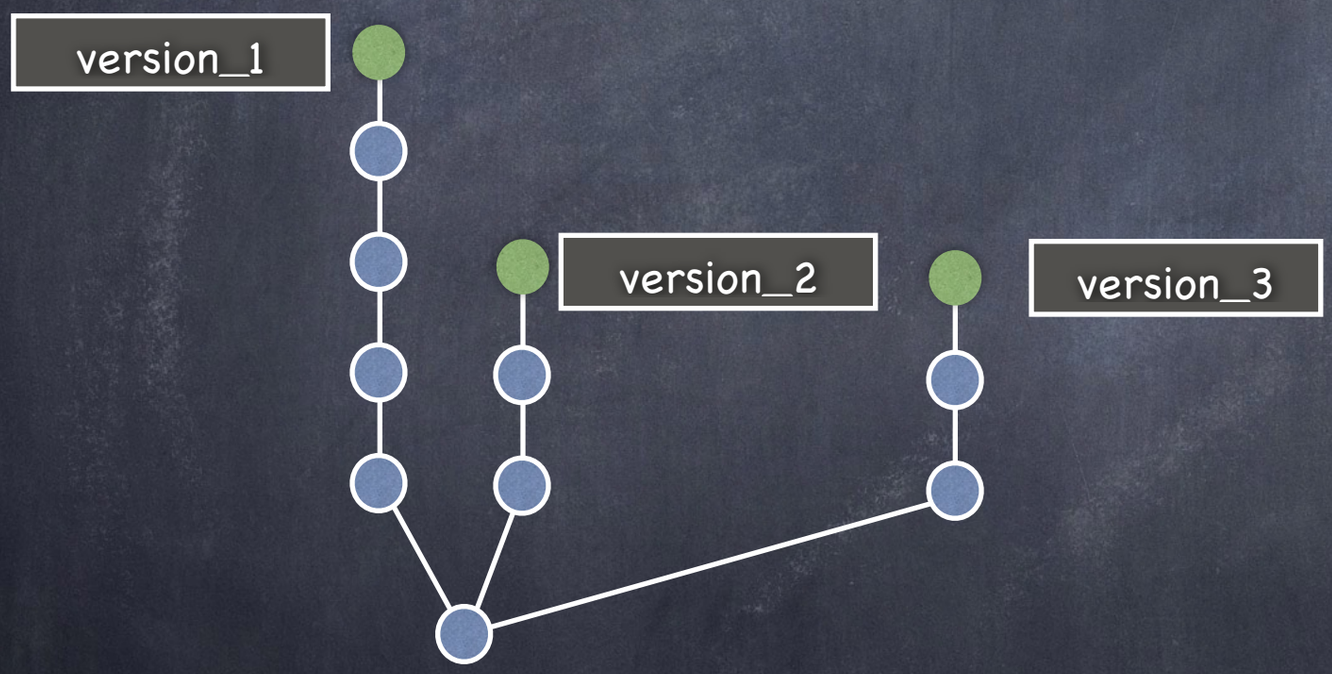
```
git checkout version_2
git cherry-pick <commit hash>
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-cherry-pick

```
git checkout version_3
git cherry-pick <commit hash>
```



Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-cherry-pick

- https://git-scm.com/docs/git-cherry-pick

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-blame

- Muestra el autor y el commit en el que cada línea de un fichero fue modificada por última vez

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-blame

```
git blame README.rst
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430  1) Acerca de
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430  2) ---------------
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430  3) Guía: "Fork de repositorios er
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430  4)
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430  5) Resumen
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430  6) ---------------
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430  7) Resumiendo un poco el proceso:
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430  8) http://aprendegit.com/fork-de-
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430  9)
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 10) 1) Ir al repositorio a colabor
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 11) 2) Realizar un fork de ese rep
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 12) 3) Clonar el fork.
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 13) 4) Hacer los cambios al fork.
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 14) 5) Subir los cambios del fork
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 15)
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 16) Resumen realizado por: Wuilmer
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 17) Twitter: @WuilmerBolivar
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 18)
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 19) Pruebas
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 20) ---------------
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 21) Prueba de modificación de un f
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 22)
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 23) Modificación de @aalbagarcia c
74b4cdd5 (Wu [redacted]    2013-11-27 23:50:10 -0430 24) practique a sincronizar sus fc
16f32149 (Gi [redacted]    2013-12-30 21:42:27 -0500 25) Modificacion hecha por @SakyaS
3b9bcc0a (En [redacted]    2014-05-07 16:29:04 -0500 26) Última prueba de pull request
4095985a (aa [redacted]    2014-06-04 20:32:31 +0200 27)
4095985a (aa [redacted]    2014-06-04 20:32:31 +0200 28) Cambio esto :)
e79209c4 (aa [redacted]    2014-06-04 20:33:11 +0200 29) y esto :P -
```

Extras: git-blame

- https://git-scm.com/docs/git-blame

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-revert

- Dado uno o más commits, este comando revierte los cambios introducidos por estos commits, generando nuevos commits revirtiendo los parches correspondientes en cada nuevo commit

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-revert

- El commit que se quiere revertir no tiene porqué ser el último de la rama activa, puede ser cualquiera
- Al intentar aplicar el parche con los cambios revertidos, esta operación puede dar lugar a conflictos.

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-revert

feature_x *

f43c00a




```
+ def method
+  #…do something
+ end
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una
Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-revert

```
git revert f43c00a
```



Este commit elimina el método

```
- def method
-   #…do something
- end
```

Este commit introduce un método

```
+ def method
+   #…do something
+ end
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-revert

- Si al aplicar el parche se producen conflictos podemos:
  - cancelar la operación y volver al estado antes de empezar la operación
    git revert --abort
  - Resolver los conflictos, añadir los ficheros con los conflictos resueltos al staging area y continuar la operación con
    git revert --continue

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-revert

- https://git-scm.com/docs/git-revert

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-filter-branch

- Se usa para reescribir grandes secciones de la historia de un proyecto
- Dado un rango de commits, este comando regenera los commits (crea commits nuevos) realizando sobre el nuevo snapshots las operaciones que se definan al ejecutarlo

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-filter-branch

- Usos más habituales:
  - Cambiar el contenido del snapshots de los commits, por ejemplo borrar un fichero que ha sido incluido en los commits del repositorio por error. Esto se consigue con la opción --tree-filter
  - Cambiar la información de los commits, por ejemplo el autor que ha realizado los commits, esto se consigue con la opción --commit-filter

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-filter-branch

- Si la historia del repositorio es larga y queremos modificar muchos commits, este comando puede resultar un poco lento
- En este caso, existe una alternativa escrita en scala "git repo cleaner" que puede acelerar el proceso: https://rtyley.github.io/bfg-repo-cleaner/

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Extras: git-filter-branch

- https://git-scm.com/book/es/v2/Git-Tools-Rewriting-History#The-Nuclear-Option:-filter-branch
- https://git-scm.com/docs/git-filter-branch

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Dropbox

- Ahora que ha finalizado el curso, y sabes cómo funciona git...

¿Crees que es una buena idea tener los repositorios de git en Dropbox?

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Dropbox

- Es una buena idea si...
  - Estás trabajando tú sólo con el repositorio en un máquina
  - Estás trabajando tú sólo en varias máquinas, y antes de hacer push/pull/fetch te aseguras que el repositorio está sincronizado completamente con Dropbox

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Dropbox

- Es una muy muy muy MALA idea si...
  - Estáis trabajando varias personas en el equipo
  - No podéis coordinaros: si dos personas hacen push simultáneamente, lo más normal es que el repositorio acabe corrompido

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Dropbox

- Si vas a trabaja tú solo:
  - Creas un repositorio bare en Dropbox:

```
git init --bare ~/Dropbox/miproyecto_bare.git
```

  - Clonas este repositorio, de forma que usas la carpeta de Dropbox como remoto

```
git clone ~/Dropbox/miproyecto_bare.git miproyecto
```


Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Buenas prácticas

- Usa ramas locales
- No dejar los mensajes de los commits vacíos
- Usar mensajes descriptivos que indiquen qué contiene el commit. ¿Qué quiere decir "trabajando"? ¿Y "..."?
- Si utilizas un sistema de tareas o bug tracking que se enlaza con git (como Jira o redmine) enlaza tus commits con tus tickets, bugs o tareas
- Haz commits pequeños. Haz commit a menudo.
- Usa ramas locales
- Atomiza los commits todo lo que puedas. Intenta no mezclar en un mismo commit código de cosas que no tengan nada que ver entre sí.
    - Por ejemplo: si has hecho tres vistas (tabla de registros, detalle de registro y formulario de alta de registro) haz tres commits, no hagas sólo uno.
- El flujo TDD es escribir test - rojo - escribir código - verde - commit.
    - Si luego te quedan muchos commits, siempre puedes hacer squashing.
- Usa ramas locales y utiliza rebase para incorporar los cambios de las otras ramas.
- Haz squashing de tus commits para eliminar aquellos que sean irrelevantes para el resto del equipo
- Evita sobreescribir la historia del repositorio (a no ser que sepas lo que estás haciendo)
- Por cierto, usa ramas locales

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Ejercicios prácticos: trabajo en equipo con merge

- Crear una página web HTML con los compañeros.
- Cada persona tendrá una página con su nombre (por ejemplo, alfonso.html)
- La web tendrá un index.html que incluirá los enlaces a todas las páginas de los componentes del equipo
- Utilizar para ello un repositorio único en el que todo el equipo tiene permiso de escritura
- Realizar integración del desarrollo usando merges

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Ejercicios prácticos: trabajo en equipo con rebase

- Crear una página web HTML con los compañeros.
- Cada persona tendrá una página con su nombre (por ejemplo, alfonso.html)
- La web tendrá un index.html que incluirá los enlaces a todas las páginas de los componentes del equipo
- Utilizar para ello un repositorio único en el que todo el equipo tiene permiso de escritura
- Realizar integración del desarrollo usando rebase

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Ejercicios prácticos: pull request

- Crear una página web HTML con los compañeros. Cada persona tendrá una página con su nombre (por ejemplo, alfonso.html). La web tendrá un index.html que incluirá los enlaces a todas las páginas de los componentes del equipo
- El equipo dispondrá de un repositorio maestro. Los miembros del equipo no tendrán permiso de escritura a este repositorio
- Cada miembro hará un "fork" del repositorio maestro y trabajará en su form
- Realizar integración del desarrollo usando pull request de github / bitbucket / gitlab

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

¡Gracias!

cursodegit.com

Diapositivas del curso "Entendiendo git" porcursodegit.com se encuentra bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

aprendegit.com