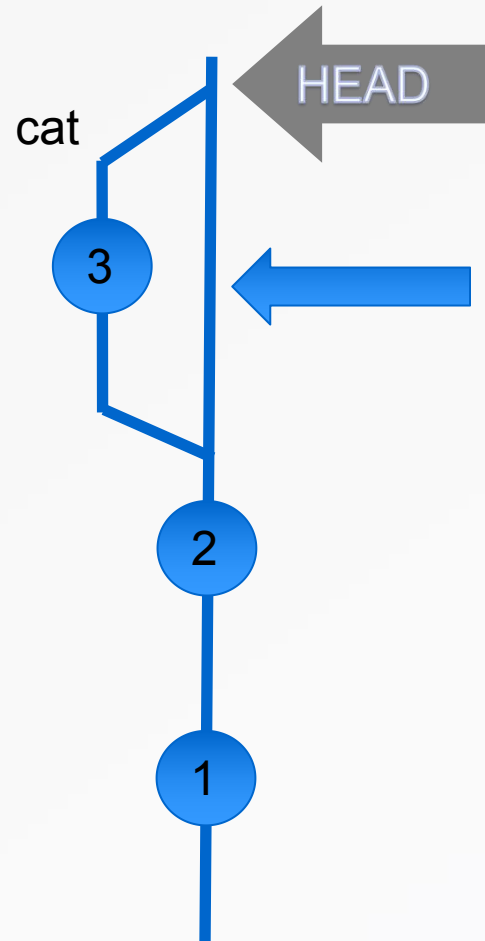


UT 2.5 GIT. Escenarios de trabajo colaborativo

ENTORNOS DE DESARROLLO

Escenario 1: Fusión de ramas sin conflictos

- Como hemos visto en la anterior sección, a la hora de fusionar una rama en la rama master se puede realizar de modo **Fast-Forward**



No ha pasado nada desde la creación del branch “cat”.

Para GIT es muy fácil integrar la rama cat en master

Escenario 1: Fusión de ramas sin conflictos

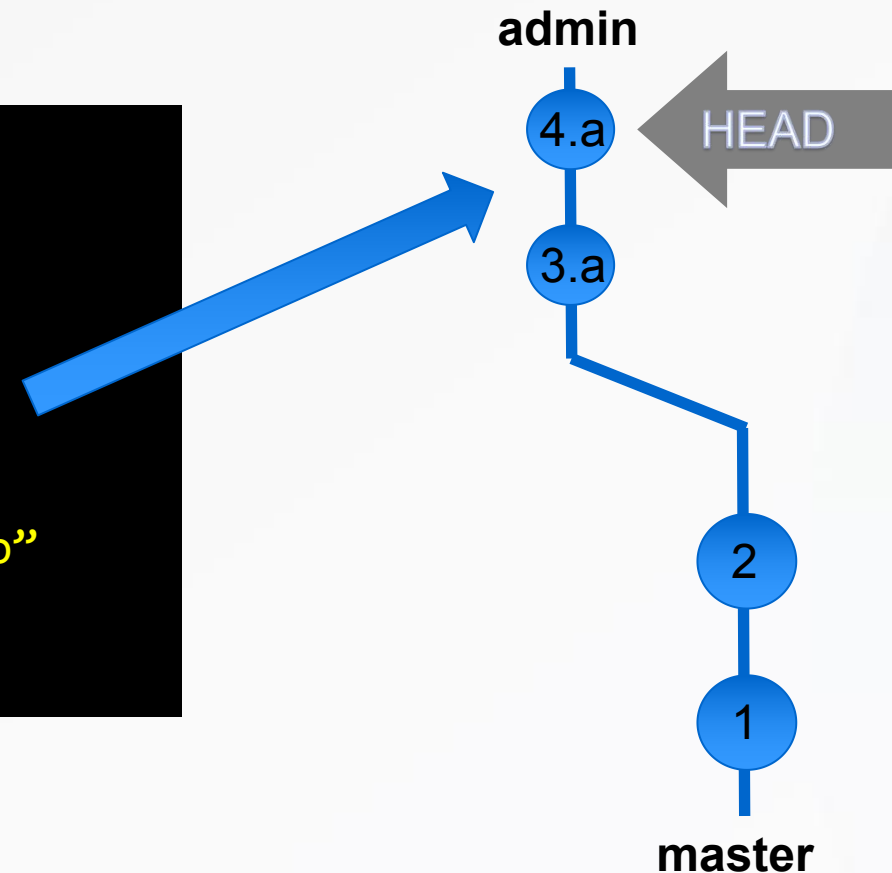
- Pero lo habitual es que en un **trabajo en grupo** y que a la hora de fusionar ocurran situaciones especiales o conflictos que habrá que resolver.
- Para ver esto de manera práctica, suponemos que vamos a un primer escenario y es que queremos **implementar una nueva funcionalidad de administración en la web**. Creamos una nueva rama **admin**.



Escenario 1: Fusión de ramas sin conflictos

- Creamos y nos movemos a la rama **admin** y confirmamos un par de html necesarios para esa nueva funcionalidad

```
$ git checkout -b admin  
Switched to a new branch 'admin'  
  
$ git add admin/dashboard.html  
$ git commit -m "Dashboard añadido"  
  
$ git add admin/users.html  
$ git commit -m "Usuario admin añadido"
```



Escenario 1: Fusión de ramas sin conflictos

- En ese mismo momento recibimos un mensaje urgente de nuestro jefe diciendo que hay errores en la rama master y que es urgente repararlos.
- Dejamos entonces nuestro trabajo en la rama “admin” y nos movemos a la rama “master” para arreglar esos errores.

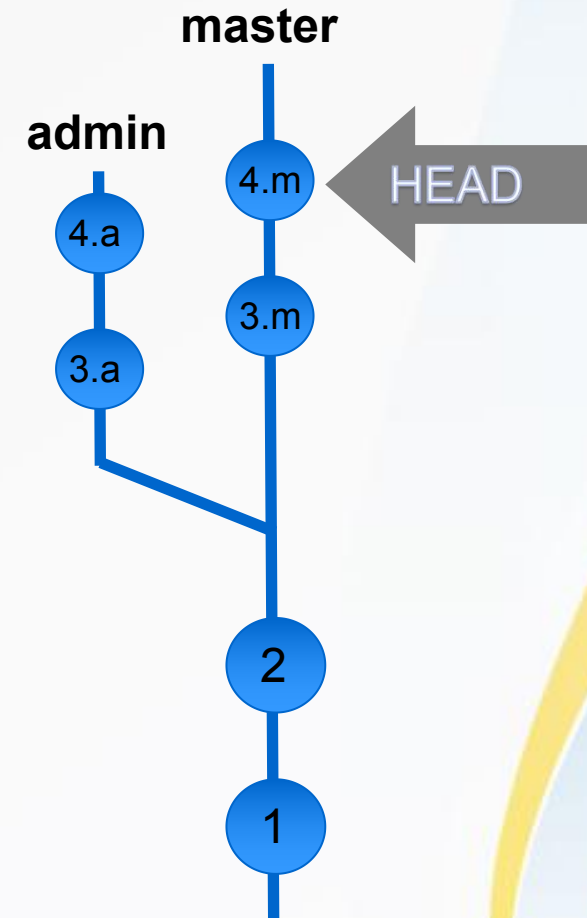
```
$ git checkout master
Switched to branch 'master'

$ git branch
  admin
* master
$ git pull
...
```

Escenario 1: Fusión de ramas sin conflictos

- Modificamos los archivos correspondientes que provocaban el error y los confirmamos

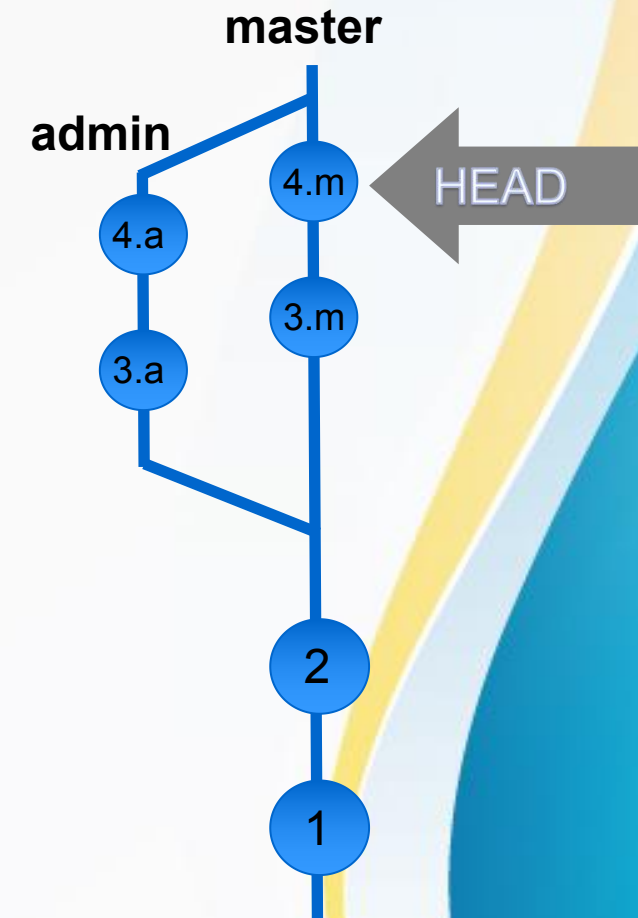
```
$ git add store.rb  
$ git commit -m 'Corrijo error de store'  
...  
$ git add product.rb  
$ git commit -m 'Corrijo error de product'  
...  
$ git push
```



Escenario 1: Fusión de ramas sin conflictos

- Solucionado todo. Volvemos a nuestra rama admin para terminar la nueva funcionalidad y nos volvemos a cambiar a admin para realizar el fusión (merge)

```
$ git checkout admin  
Switched to branch 'admin'  
  
.....  
  
$ git checkout master  
Switched to branch 'master'  
  
$ git merge admin
```



Escenario 1: Fusión de ramas sin conflictos

- Y de repente aparece Vi, el editor de texto por defecto de Git en Linux
- Esto ocurre porque desde la creación de la rama **admin** hasta su fusión a **master**, en ésta ha habido cambios.

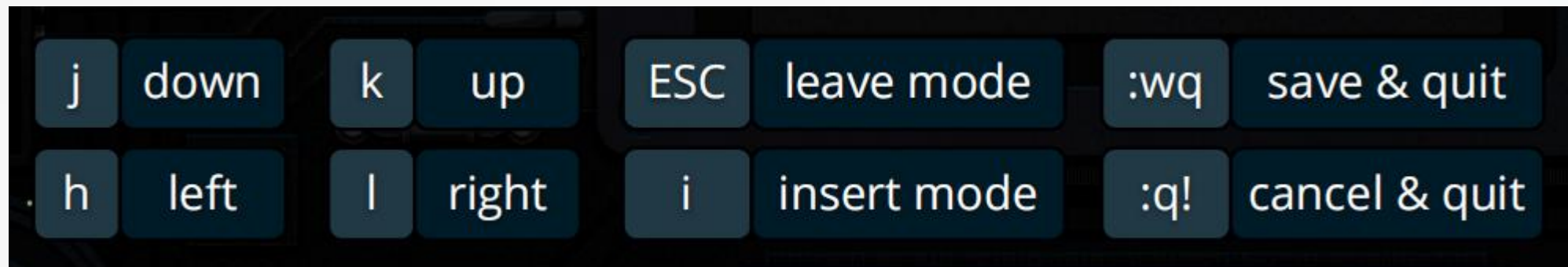
```
Merge branch 'admin'

# Por favor ingrese un mensaje de commit que explique por qué es necesaria esta fusión,
# especialmente si esto fusiona un upstream actualizado en una rama de tópico.
#
# Líneas comenzando con '#' serán ignoradas, y un mensaje vacío aborta
# el commit.
```

- Desde este editor podemos modificar el mensaje que quedará guardado tras este “merged”.

Escenario 1: Fusión de ramas sin conflictos

- Para mantener este texto por defecto, pulsaremos las teclas **:wq** y presionaremos **Enter**.
- Si se desea modificar este mensaje por defecto, estos son los diferentes comandos que podremos necesitar para modificarlo:



Escenario 1: Fusión de ramas sin conflictos

- En nuestro caso, GIT ha realizado una fusión recursiva (recursive merge).
- Cuando esto ocurre, GIT crea un nuevo commit que registra la fusión de ambas ramas.
- Este commit no va asociado a ningún fichero, simplemente a la fusión realizada.

```
$ git log
commit 57432439aef324db150188b58d0b37352883bd969dcf (HEAD -> master, origin/master)
Author: Roberto Hidalgo <robertohidalgo.informatica@iespaconolla.es>
Date: Thu Oct 24 21:56:07 2019 +0200

    Merge branch 'admin'
```

Escenario 1: Fusión de ramas sin conflictos

- La fusión vista no implica ningún conflicto ya que en **master** se crearon/modificaron ficheros distintos a los que se crearon/modificaron en **admin**.
- En realidad aún no hemos trabajado en equipo con otro desarrollador del equipo y además lo hemos hecho todo a nivel local.
- Vamos a ver un par de escenarios más, ya teniendo en cuenta a dos desarrolladores.

Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- De nuevo tenemos una desarrolladora llamada Jane que quiere empezar a desarrollar una nueva funcionalidad, así que volverá a clonar el repositorio



```
$ git clone https://github.com/robertohidalgo/cutreweb.git
```

Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- Jane comienza a hacer cambios, añade dos nuevos ficheros a nuestra web

```
jane $ git status
On branch master
Your branch is up to date with 'origin/master'.

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

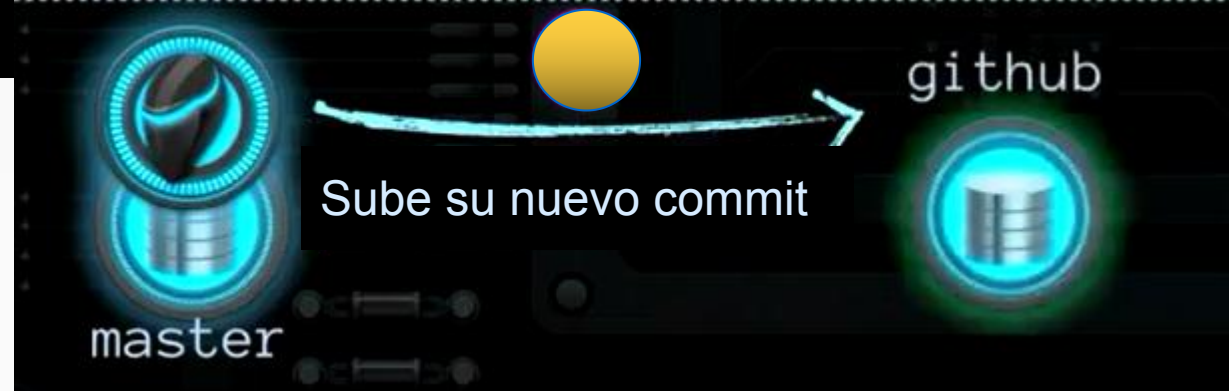
nothing added to commit but untracked files present (use "git add" to track)

jane $ git add --all
jane $ git commit -m "Añadida página clientes y productos"
[master 0b65132] Añadida página clientes y productos
 2 files changed, 2 insertions(+)
 create mode 100644 clientes.html
 create mode 100644 productos.html
```

Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- Jane sube al remoto asociado a su repositorio local sus cambios, es decir, a GitHub.

```
jane $ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 432 bytes | 432.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/robertohidalgo/cutreweb.git
5792f9a..0b65132 master -> master
```



Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- Tenemos a otro desarrollador, Gregg, que sigue trabajando sobre el proyecto, hace un cambio sobre un fichero ya existente, por ejemplo, sobre *index.html*

```
gregg $ git commit -a -m "Añado nuevo enlace sobre index"
[master 7865ac4] Añado nuevo enlace sobre index
1 file changed, 3 insertions(+), 2 deletions(-)
```

- La situación actual es que lo el último commit en GitHub y el que acaba de hacer Gregg son diferentes



Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- Gregg como desconoce que hay un nuevo commit en GitHub, realiza la operación pertinente para incluir estos nuevos cambios en GitHub, pero....

```
gregg $ git push
To https://github.com/robertohidalgo/cutreweb.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/robertohidalgo/cutreweb.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

- Es **rechazado** por que no puede escribir sobre el commit que realizó Jane.

Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- La solución sería actualizar nuestro repositorio local a la última versión en GitHub, y después volver a realizar la orden anterior.

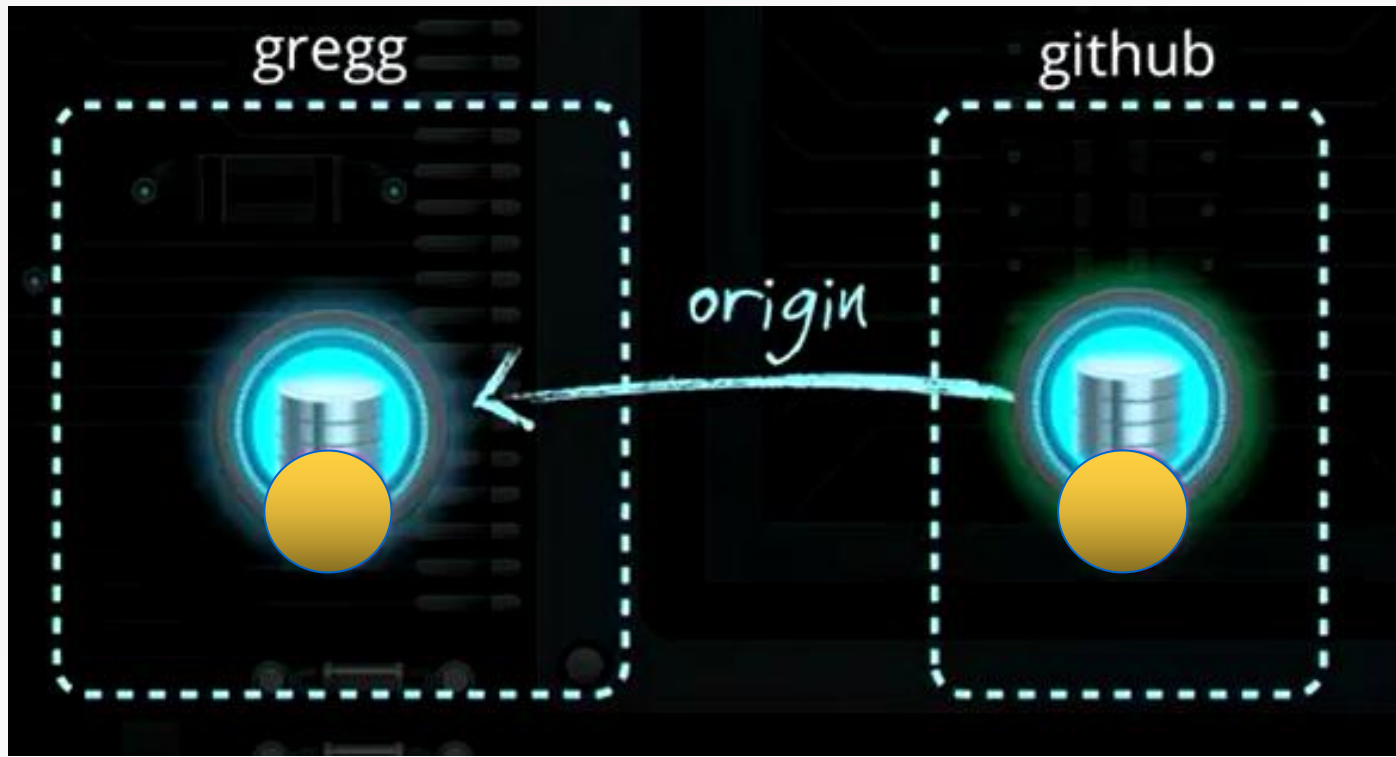
```
gregg $ git pull
....
gregg $ git push
....
```

- Pero, ¿qué se está haciendo exactamente?

Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

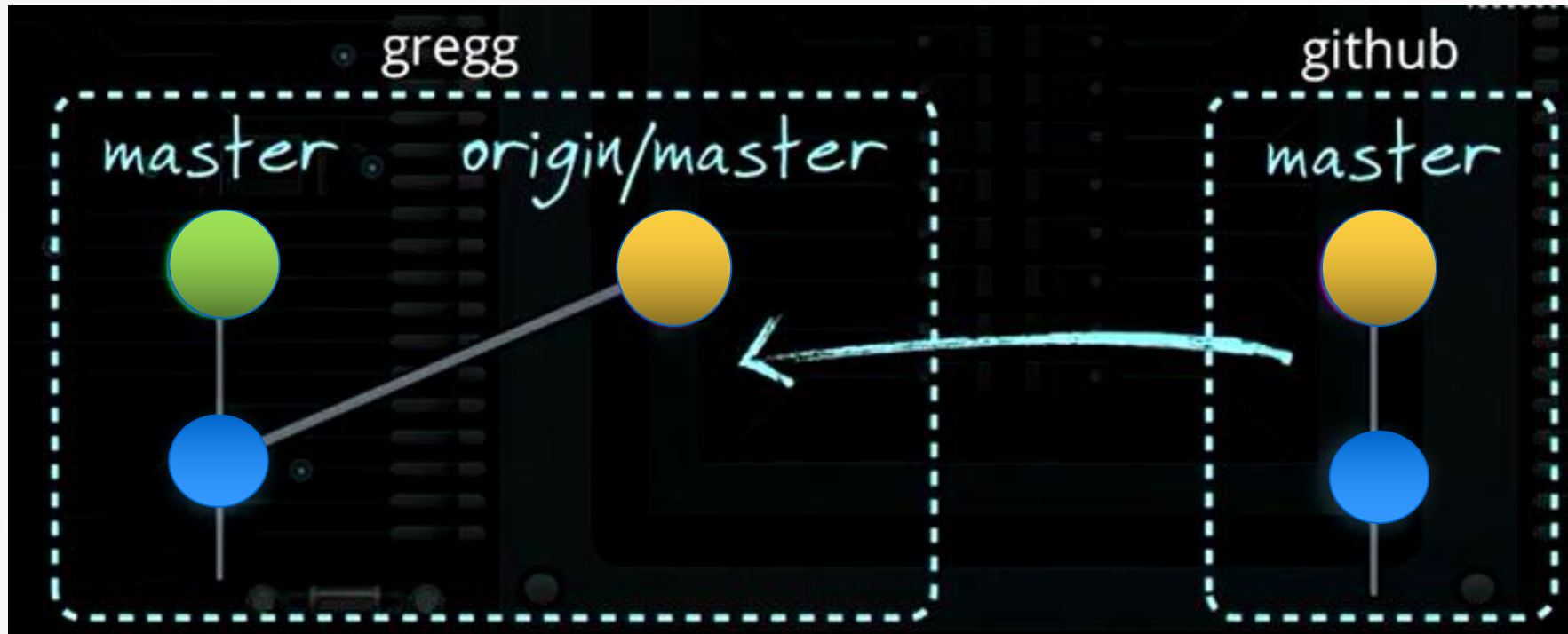
- `$ git pull` (1er paso)

Sincroniza nuestro repositorio local con el remoto



Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- Si queremos realizar únicamente este primer paso se puede optar por ejecutar `$ git fetch`. Hay que tener en cuenta que nuestro repositorio local NO se ha actualizado aún



Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- `$ git pull` (2º paso)

Fusiona la rama oculta **origin/master** con **master**.

- Existe otra instrucción en git para hacer esta acción específica y es `$git merge origin/master`

En resumen:

```
$ git pull es lo mismo que hacer:  
$ git fetch  
$ git merge origin/master
```

Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- Es importante mencionar que en este tipo de situaciones, también aparecerá el editor de texto Vi, para indicarnos el mensaje que se asociará a esta fusión implícita que se produce al hacer `$ git pull`
- En este caso, el texto por defecto sería:

```
1 Merge branch 'master' of https://github.com/robertohidalgo/cutreweb
2 # Please enter a commit message to explain why this merge is necessary,
3 # especially if it merges an updated upstream into a topic branch.
4 #
5 # Lines starting with '#' will be ignored, and an empty message aborts
6 # the commit.
```

Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- Si salimos del editor guardando cambios habremos realizado el **\$git pull** con éxito y por tanto habremos fusionado la rama oculta **origin/master** con nuestra rama **master**

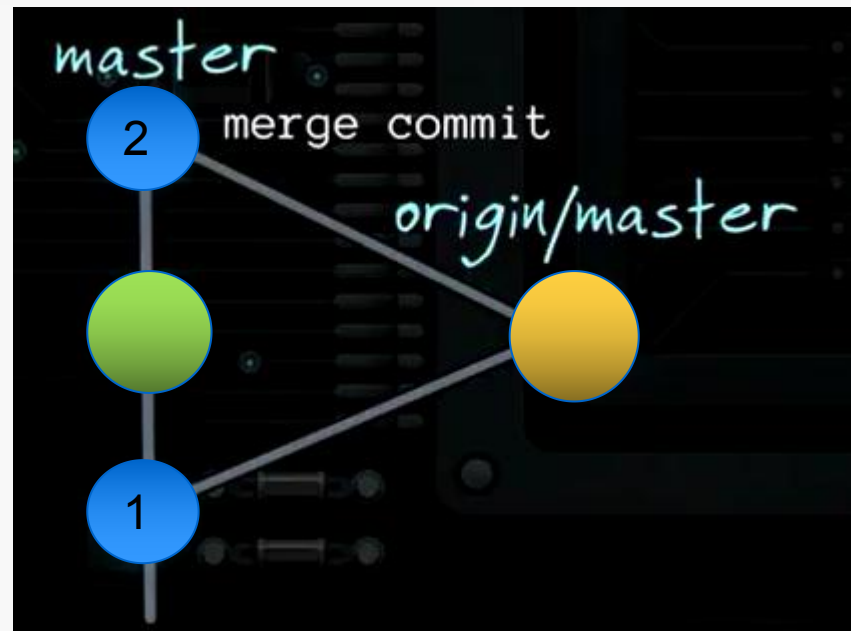
```
gregg $ git pull
```

```
remote: Enumerating objects: 5, done.  
remote: Counting objects: 100% (5/5), done.  
remote: Compressing objects: 100% (2/2), done.  
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reuse 0  
Unpacking objects: 100% (4/4), done.  
From https://github.com/robertohidalgo/cutres  
   5792f9a..0b65132  master    -> origin/master  
Merge made by the 'recursive' strategy.  
 clientes.html | 1 +  
 productos.html | 1 +  
 2 files changed, 2 insertions(+)  
 create mode 100644 clientes.html  
 create mode 100644 productos.html
```

De nuevo una
fusión recursiva
(recursive merge).

Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

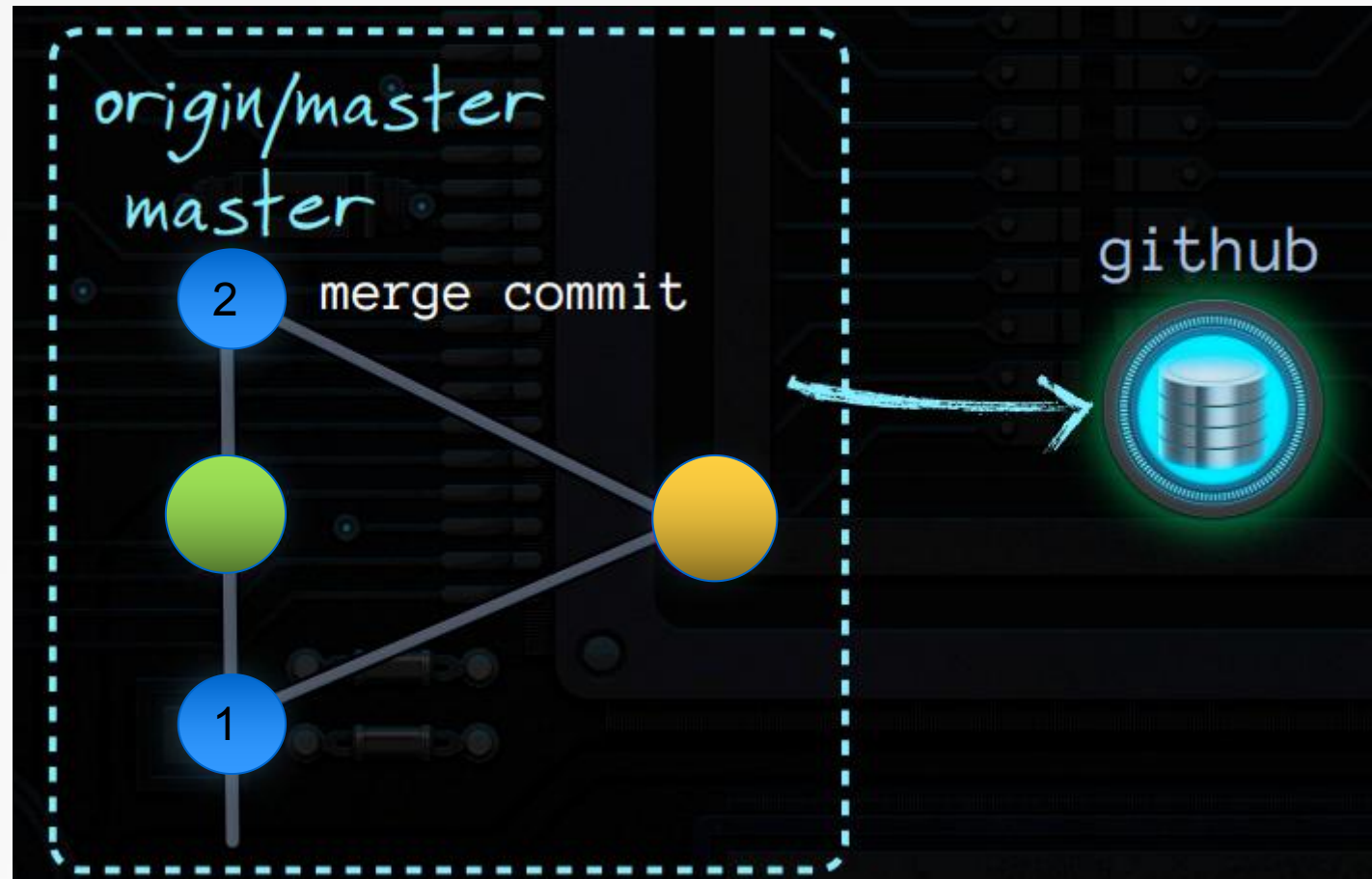
- Si echamos un vistazo a como queda nuestra línea del tiempo vemos que tras hacer **\$ git pull** queda así:



origin/master no sabe nada acerca de los nuevos cambios de Gregg

Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

- La actualización de origin/master se producirá una vez que Gregg ejecute **\$ git push**



Escenario 2: Rechazo al subir cambios locales sobre remoto modificado

```
gregg $ git log
```

```
commit ea15815797eadff53c11b56a36c06461b866e168 (HEAD -> master, origin/master, origin/HEAD)
```

```
Merge: 7865ac4 0b65132
```

```
Author: Gregg <gregg@gregg.com>
```

```
Date: Sun Nov 3 19:43:42 2019 +0100
```

```
Merge branch 'master' of https://github.com/robertohidalgo/cutreweb
```

```
.....
```

Escenario 3: Resolución de conflictos

- De nuevo Gregg y Jane, están trabajando sobre el mismo código, al mismo tiempo. Esta vez, ambos están modificando el mismo fichero, en concreto, **index.html**, en concreto el title, uno ha añadido la palabra PROFESIONAL, mientras que JANE ha metido la palabra SEMIPROFESIONAL.

Gregg

```
GNU nano 4.3 index.html Modified
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>EJEMPLO MAQUETACIÓN PROFESIONAL EN HTML5</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
```

Jane

```
GNU nano 4.3 index.html Modified
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>EJEMPLO MAQUETACIÓN SEMIPROFESIONAL EN HTML5</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
```

Escenario 3: Resolución de conflictos

- De nuevo Jane hace **commit** y **push** sobre el repositorio GitHub, mientras que Gregg simplemente ha hecho commit sobre su repositorio local



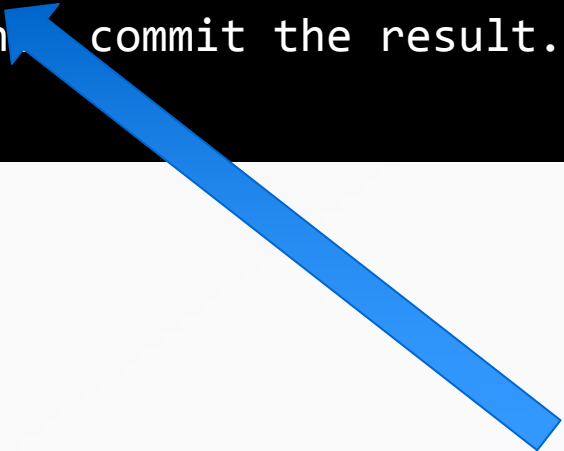
Escenario 3: Resolución de conflictos

- Gregg intenta hacer push, pero se le vuelve a **rechazar**.
- Según lo que aprendió del escenario anterior, para poder subir sus cambios a GitHub debe hacer nuevo **git pull** antes de **git push**.

Escenario 3: Resolución de conflictos

- Gregg hace git pull y...

```
gregg $ git pull
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```



NO OLVIDAR
editar este fichero

Escenario 3: Resolución de conflictos

- Cuando Gregg edita index.html se encuentra esto:

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<<<<<<< HEAD
<title>EJEMPLO MAQUETACIÓN PROFESIONAL EN HTML5</title>
=====
<title>EJEMPLO MAQUETACIÓN SEMIPROFESIONAL EN HTML5</title>
>>>>>>> 94b36c3b142efb76df1fe369956f4c088a7c2c1a
<link rel="stylesheet" href="css/estilo.css">
```

- Gregg debe elegir, cuál de los dos cambios es el correcto, y se queda con el suyo.

Escenario 3: Resolución de conflictos

- Gregg deja el fichero así:

```
GNU nano 4.3 index.html
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>EJEMPLO MAQUETACIÓN PROFESIONAL EN HTML5</title>
<link rel="stylesheet" href="css/estilo.css">

</head>
```

- Tras esta modificación, Gregg ejecuta:

```
gregg $ git commit -a
```

- Volviendo a aparecer Vi, esta vez para informar del mensaje del merge a realizar y la lista de ficheros que se han reparado para evitar el conflicto.

Escenario 3: Resolución de conflictos

```
Merge branch 'master' of https://github.com/robertohidalgo/cutreweb
```

```
# Conflicts:
#       index.html
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#       .git/MERGE_HEAD
# and try again.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch and 'origin/master' have diverged,
# and have 1 and 1 different commits each, respectively.
#   (use "git pull" to merge the remote branch into yours)
#
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#       modified:   index.html
#
```


Escenario 3: Resolución de conflictos

