educación IT
CODE YOUR FUTURE

Git:
Desarrollo colaborativo

Módulo 5

Configuración avanzada

Configuración avanzada

Antes de terminar con el curso, veamos parámetros de configuración adicionales en Git que podrían llegar a hacernos el trabajo un poco más fácil.

Alias

Una de las capacidades más utilizadas por los usuarios de Git en la línea de comandos es la capacidad de **crear alias** con el comando:

```
> git alias
```

Este comando permite **crear atajos** en los momentos en que **no se desea escribir un comando entero** porque, por ejemplo, se utiliza constantemente.

Puede ser el caso de **`git log --oneline --all --graph`**. Es posible guardarlo en un alias de la siguiente manera:

```
git config --global alias.lo "log --oneline
--graph --all"
```

De esta manera, la próxima vez que tengamos que ejecutar ese comando podemos simplemente hacer esto:

```
git log

* 8d86da1 (HEAD -> restauracion) Add copy change
* 7737d6b add new change
| * 93aaac9 (origin/dependabot/bundler/) Bump activesupport in
rendering-data-as-graphs
|/
*   297762a (origin/master, origin/HEAD, master) Merge pull request
|\
```

Plantilla de commits

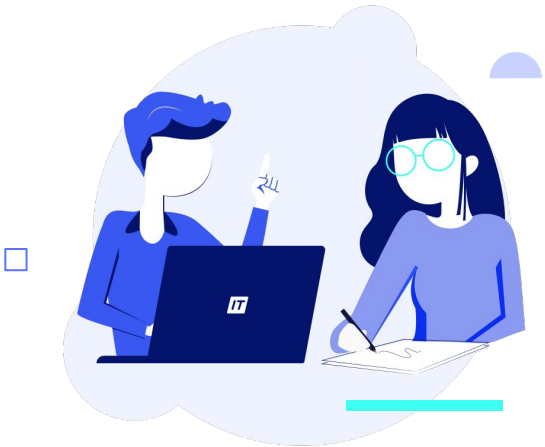
Podemos utilizar también una configuración a veces poco conocida que nos permite agregar **plantillas personalizadas a los mensajes del comando `commit`** utilizando el siguiente parámetro de configuración:

```
> git config --global commit.template ~/.mensajecommit.txt
```

De esta manera le estamos diciendo a Git que cada vez que se ejecute el comando **`git commit`**, muestre además el texto que está guardado en ese archivo.

Esto es muy beneficioso ya que se pueden unificar maneras de trabajo sin tener que acordarnos constantemente de ellas. Por ejemplo, si se desea que en cada **`commit`** se esté dejando determinado mensaje en tal o cual formato.

Esto nos sirve no solamente a nosotros, sino también especialmente si estamos trabajando con un equipo de personas.



Expiración del Reflog

Tenemos que tener en cuenta que el **`reflog`** por defecto viene configurado para que tenga una **duración máxima de 30 días para casos en los que la expiración sea del tipo *unreachable* y 90 días para casos normales.**

Sin embargo, podemos **extenderlo o disminuirlo según nuestra necesidad** en cada repositorio con el siguiente parámetro de configuración:

```
> git config gc.reflogExpire
> git config gc.reflogExpireUnreachable
```

Se debe tener en cuenta que la diferencia entre casos normales y los “unreachable” es básicamente la capacidad que tenemos para acceder a referencias desde nuestro árbol.

Ejemplo

```
              1234567 <-- master
                /
...--9999999--8888888 [master@{1}]
                \
                 \ 3333333 [master@{2}]
```

Podemos observar que si estuviéramos parados, por ejemplo, en el *branch master* y fuéramos para atrás en relación a nuestro *log* nos toparemos con el *`commit 888888`*… o, en otras palabras, *`master@{1}`*. Este `commit` es accesible de momento, va a estar disponible en el `reflog` hasta 90 días.

Si fuéramos un paso más atrás en el historial, nos vamos a topar con el *`commit 9999999`*… y así sucesivamente.

Si prestamos atención al *`commit 333333`*…, vemos que no se encuentra disponible desde nuestro árbol directo, entonces es *unreachable* y permanecerá disponible por un plazo de 30 días.

¡Sigamos trabajando!