## La segunda confirmación

Para ver el verdadero poder del control de versiones, necesitamos hacer un cambio en el proyecto y confirmar ese cambio. Aquí sólo añadiremos otra línea a *hello\_git.py*:

**hello\_git.py**

print("Hello Git world!")  
print("Hello everyone.")

Cuando comprobemos el estado del proyecto, veremos que Git se ha dado cuenta del archivo que ha cambiado:

git\_practice$ git status  
❶ On branch main  
Changes not staged for commit:  
 (use "git add <file>..." to update what will be committed)  
 (use "git restore <file>..." to discard changes in working directory)  
  
❷ modified: hello\_git.py  
  
❸ no changes added to commit (use "git add" and/or "git commit -a")  
git\_practice$

Vemos la rama en la que estamos trabajando ❶, el nombre del archivo que se modificó ❷, y que no se ha confirmado ningún cambio ❸. Confirmemos el cambio y comprobemos de nuevo el estado:

❶ git\_practice$ git commit -am "Extended greeting."  
[main 945fa13] Extended greeting.  
 1 file changed, 1 insertion(+), 1 deletion(-)  
❷ git\_practice$ git status  
On branch main  
nothing to commit, working tree clean  
❸ git\_practice$ git log --pretty=oneline  
945fa13af128a266d0114eebb7a3276f7d58ecd2 (HEAD -> main) Extended greeting.  
cea13ddc51b885d05a410201a54faf20e0d2e246 Started project.  
git\_practice$

Hacemos un nuevo commit, pasando la bandera -am cuando utilicemos el comando git commit ❶. La bandera -a indica a Git que añada todos los archivos modificados en el repositorio a la confirmación actual. (Si creas algún archivo nuevo entre confirmaciones, vuelve a emitir el comando git add . para incluir los archivos nuevos en el repositorio). La bandera -m indica a Git que registre un mensaje en el registro para esta confirmación.

Cuando comprobamos el estado del proyecto, vemos que volvemos a tener un árbol de trabajo limpio ❷. Por último, vemos las dos confirmaciones en el registro ❸.

[anterior](b04_11.html)[Subtema 12 de 16: (Ver todo)](b04.html)[siguiente](b04_13.html)