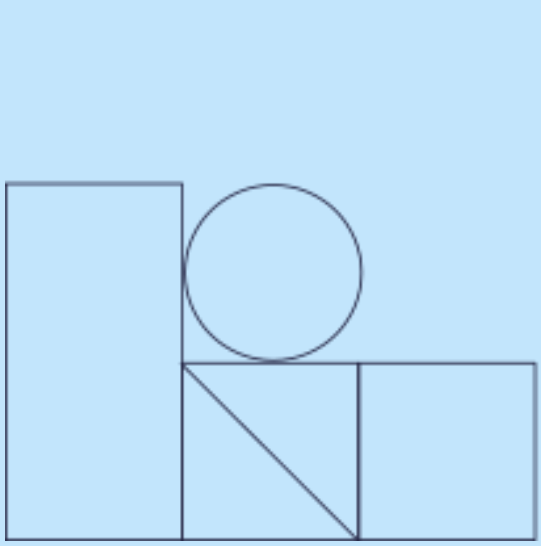
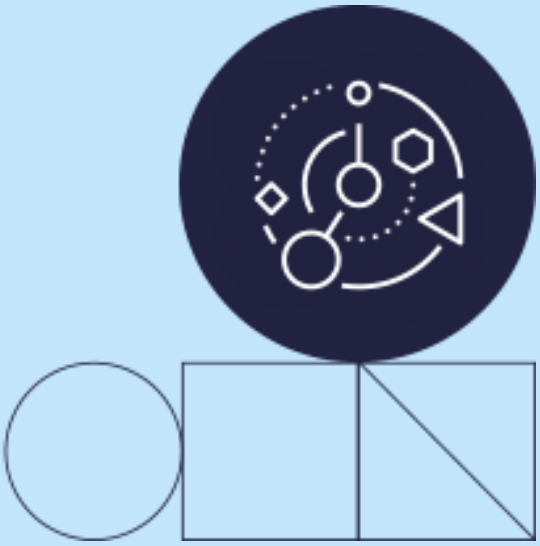
IBM SkillsBuild

IBM

Git y Github

Commits





Índice

Definición

El comando *git commit* captura una instantánea de los cambios realizados actualmente en nuestro proyecto. Las instantáneas creadas con *commit* se pueden considerar como versiones "seguras" de un proyecto. *Git* nunca las cambiará a menos que se lo solicitemos explícitamente.

Antes de la ejecución de *git commit*, se usa el comando *git add* "preparar" cambios en el proyecto que se almacenarán en nuestro repositorio.

Estos dos comandos *git commit* y *git add* son dos de los más utilizados.

Uso de commit

En el anterior tema habíamos dejado nuestro archivo en el *stage área*.

Para hacer una instantánea de su contenido usaremos el comando "*commit*".

La primera vez que hacemos un "*commit*" se nos pedirá nuestro mail y nombre.

Git sólo pedirá estos datos la primera vez que lo usemos en un equipo:

```
$ git commit -m "Version 1.0.0"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

Una vez introducidos los datos siguiendo las indicaciones que nos muestra *GIT*, volvemos a ejecutar el "*commit*". Deberemos añadir también una descripción para la instantánea, en nuestro caso le pondremos, por ejemplo, "Version 1.0.0".

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git commit -m "Version 1.0.0"
[master (root-commit) 00350f8] Version
1.0.0
 1 file changed, 0 insertions(+), 0
deletions(-)
 Créate mode 100644 index.html
```

Ya tenemos guardada una primera instantánea de nuestro proyecto, es decir, una copia de nuestro código tal y como está en el momento actual.

Si más adelante necesitásemos volver a la versión del proyecto tal y como estaba en el momento actual, podríamos recuperarla.

Si ahora hacemos un "git status -s":

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s

?? estilo.css
?? main.js
```

Es la carpeta que usará *GIT* para crear las áreas de ensayo local y de repositorio.

Veremos que nuestro archivo ha desaparecido del listado. Esto se produce porque en el momento en que se agrega un archivo al repositorio y ya hay una copia de respaldo, el archivo deja de estar en el *stage área* y, por lo tanto, ya no nos lo mostrará. Con el comando "*git status -s*" sólo obtendremos información de aquello que no está en el repositorio.

Si hiciésemos algún cambio en el contenido de nuestro archivo y ejecutásemos de nuevo un "*git status -s*":

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s

M index.html
?? estilo.css
?? main.js
```

Veríamos que ahora sí que nos informa del estado. Concretamente la M nos indica que el archivo ha sido modificado y tiene cambios que no han sido respaldados aún.

Si quiero hacer un respaldo del archivo con las modificaciones nuevas tendremos que poner el archivo en el área de ensayo y ejecutar el comando:

git commit -m "nueva descripción"

Es decir, que para cada cambia que hagamos, tenemos que pasar de nuevo por el *stage área* (con lo que lo veríamos con una M) y luego hacer un *commit*.

Stage área:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git add index.html
```

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s

M index.html
?? css/
?? js/
```

Veremos que ahora sale una A que nos indica que está haciendo el seguimiento del archivo index.html. Dicho archivo ahora está en el *staging área*, el área de ensayo.

Commit:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git reset –hard 093f1c5

HEAD is now at 093f1c5 Version 1.0.0
```

Llegado este punto tendríamos almacenadas dos instantáneas de nuestro archivo index.html.

Para sacar un listado de todas las copias que tenemos en el repositorio local usaremos el comando:

git log –oneline

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git log oneline

934978a (HEAD -> master) Version 1.0.1
093f1c5 Version 1.0.0
```

Como vemos, en nuestro caso tenemos dos versiones del archivo, la V1.0.0 y la V1.0.1

Si quisiésemos restaurar el archivo a la primera versión que teníamos, es decir, hacer una restauración del archivo, deberíamos usar el comando:

git reset –hard 093f1c5

Como vemos, hay que pasarle el código alfanumérico de la instantánea que queremos recuperar.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git commit -m “Version 1.0.1”
[master 934978a] Version 1.0.1
 1 file changed, 1 insertion(+), 1
deletion(-)
```

Con esto ya tendríamos nuestro archivo *index.html* restaurado a la primera versión.

Hay que señalar que, al haber retrocedido a la primera instantánea que teníamos (la V1.0.0), la segunda versión de mi archivo (la V1.0.1) ya no existe, no está disponible, desaparece. Es decir, podemos volver atrás a una versión anterior, pero una vez hecho esto perderemos todas las versiones posteriores a dicha versión.

Hemos visto cómo hacer un seguimiento a un archivo en concreto.

Para hacer un seguimiento a todos los archivos de nuestro proyecto usaremos el comando:

git add .

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git add .
```

Con esto habremos añadido todos mis archivos al *stage area*.

Si hacemos un git status -s veremos que Git le está haciendo seguimiento a todos los archivos de nuestro directorio de trabajo.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s

A  estilo.css
A  main.js
```

Si ahora hacemos cambios en el contenido de los archivos *index.html* y *estilo.cs*, al guardar dichos cambios y hacer un git status -s

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s
```

Veremos que Git nos indica con M los archivos que hemos modificado.
Para hacer una nueva instantánea del proyecto incluyendo los últimos cambios realizados, deberíamos hacer dos pasos:

- Hacer un add para que vuelva a añadir ambos archivos al *staging área*
- Hacer un commit para que nos pase los archivos a nuestro repositorio.

En casos como este, en el que tenemos que hacer un add y un commit, podemos hacer ambas operaciones en un solo paso, usando el comando:

git commit -am “descripción”

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git commit -am “Version 1.0.2”

[master fd8ada9] Version 1.0.2
 3 files changed, 5 insertions(+)
 create mode 100644 estilo.css
 create mode 100644 main.js
```

Si hacemos ahora un status no debería aparecer nada porque ya están todos los archivos agregados al repositorio.

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git status -s

AM  estilo.css
 M  index.html
A   main.js
```

Llegado este punto tenemos varias instantáneas de nuestro código en el repositorio local, es decir, en nuestro equipo. Pero no están subidas a la nube.

Para ello, usaremos más adelante GitHub.