3ºB GITT + BA

Práctica 1: Instalación de herramientas

Alba Arias Lamas

29 de enero de 2023

Objetivos

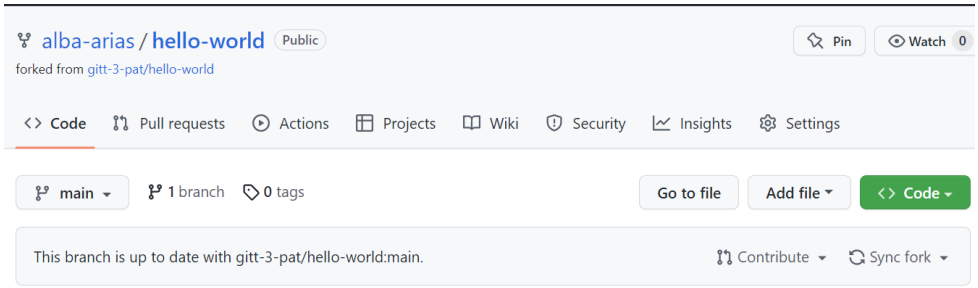
Esta práctica tiene como objetivo proporcionarnos con nociones básicas de Git como sistema de control de versiones de código fuente de Software, así como de GitHub como plataforma de desarrollo.

Tanto Git como cualquier otro sistema de control de versiones permiten gestionar y registrar los cambios que se hacen sobre un proyecto, guardando tanto la versión más reciente como las anteriores por si fuera necesario volver a una versión anterior en un determinado momento.

Desarrollo

El desarrollo consistirá en probar una serie de comandos sobre un repositorio. Por ello, primero debemos asegurarnos de tener una cuenta en GitHub. La nuestra será **alba-arias.**

A continuación, haremos un fork sobre el repositorio https://github.com/gitt-3-pat/hello-world de manera que se creará un nuevo repositorio en nuestro perfil, copia del original.



Git clone

Una vez creado este nuevo repositorio procederemos a clonar nuestro repositorio (el fork) en local, cuya URL es: https://github.com/alba-arias/hello-world.git.

```
PS C:\Users\arias\Documents\PAT> git clone https://github.com/alba-arias/hello-world.git
Cloning into 'hello-world'...
remote: Compressing objects: 100% (4/4), done.
remote: Total 38 (delta 0), reused 0 (delta 0), pack-reused 34
Receiving objects: 100% (38/38), 59.56 KiB | 1.06 MiB/s, done.
PS C:\Users\arias\Documents\PAT>
```

La principal diferencia entre haber hecho un **fork** sobre un repositorio y **clonarlo** con **git clone** tal y como hemos hecho ahora mismo radica en el control que tenemos sobre dicho repositorio. Supongamos que nuestro repositorio original es X:

- **Git clone**: Supongamos que clonamos el repositorio X. Si no somos los dueños de este repositorio o no nos han otorgado derechos sobre él, no podremos incorporar en X los cambios que realicemos. Es decir, no podremos hacer **git push origin**.
- **Fork**: Creamos un repositorio nuevo Y del que somos propietarios, copia de X. Sobre este nuevo repositorio sí que podremos realizar cambios y actualizarlos sobre el propio Y, pero no se modificará X. Sin embargo, podemos solicitar la integración de nuestros cambios en el proyecto original por medio de **pull request**.

En este caso, estamos clonando nuestro propio repositorio (el fork del original), por lo que no tendremos problemas a la hora de actualizar nuestros cambios en el mismo.

Git status

A continuación se muestra el estado actual del repositorio en el que nos encontramos. Permite ver los cambios que están pendientes de ser confirmados.

```
PS C:\Users\arias\Documents\PAT\hello-world> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
PS C:\Users\arias\Documents\PAT\hello-world>
```

En este caso **git status** nos revela que nos encontramos en la rama "main" (el proyecto consta de una única rama por el momento), y que no tenemos cambios pendientes de actualizar con el repositorio original. Además, no tenemos cambios pendientes de confirmar.

Para ver cómo varía esta información al modificar el repositorio, se procederá a crear un nuevo archivo .txt denominado "modificacion.txt". Si ahora ejecutamos el comando **git status**, se nos notifica de que tenemos un archivo sin seguimiento y, evidentemente, sin confirmar.

```
PS C:\Users\arias\Documents\PAT\hello-world> git status
On branch main
Your branch is up to date with 'origin/main'.

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

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\arias\Documents\PAT\hello-world>
```

Git add

Por medio de **git add** podemos dar seguimiento a los archivos que hayan sido modificados o creados y que no tenían seguimiento anterior. Es el paso previo a la confirmación de los cambios, ya que indica que se incluirán actualizaciones en la siguiente versión del proyecto.

Tras ejecutar el comando **git add** consultamos el estado de nuestro directorio por medio de **git status** y obtenemos la siguiente información:

```
PS C:\Users\arias\Documents\PAT\hello-world> git add .
PS C:\Users\arias\Documents\PAT\hello-world> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   modificacion.txt

PS C:\Users\arias\Documents\PAT\hello-world>
```

Tal y como se puede apreciar, en estos momentos al archivo "modificacion.txt" ya se le está dando seguimiento y únicamente queda pendiente de confirmación por medio de **commit** para ser registrado como cambio.

Git commit

A continuación confirmaremos los cambios realizados en el proyecto a nivel local por medio del comando **git commit**. Ahora, esta versión actual del repositorio se puede considerar como una versión "segura" que no se cambiará ni eliminará a no ser que se solicite expresamente.

Como el cambio realizado fue la creación de un documento .txt que carece de verdadera función, la descripción de nuestro commit es "Trasteando".

```
PS C:\Users\arias\Documents\PAT\hello-world> git commit -m "Trasteando"
[main 00056aa] Trasteando
 1 file changed, 1 insertion(+)
 create mode 100644 modificacion.txt
PS C:\Users\arias\Documents\PAT\hello-world>
```

Tal y como se ha dicho antes, Git permite trabajar en proyectos sobre versiones distintas a la actual. Esto es posible gracias a que cada commit posee un identificador que nos permite distinguir las diferentes versiones que hemos confirmado de nuestro repositorio a lo largo del desarrollo del proyecto.

Git log

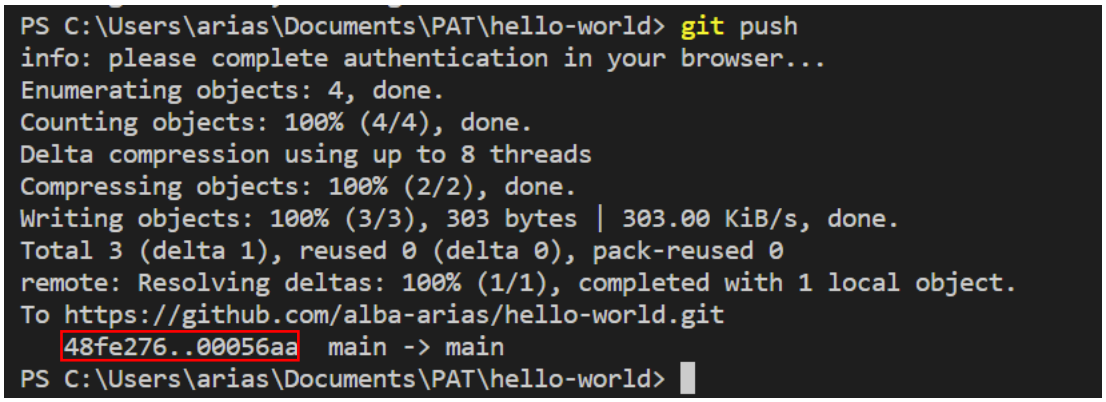
Podemos obtener estos identificadores con el comando **git log**.

```
PS C:\Users\arias\Documents\PAT\hello-world> git log
commit 00056aa4879dbc02ddbb2e26a863c7f1e0ca8511 (HEAD -> main, origin/main, origin/HEAD)
Author: Alba <202002470@alu.comillas.edu>
Date:   Sat Jan 28 17:10:54 2023 +0100

    Trasteando

commit 48fe2767a1dec83ae423cca0c0f8c92204ba6f65
Merge: 5038239 5b68377
Author: Juan Antonio Breña Moral <bren@juanantonio.info>
Date:   Mon Dec 20 19:26:31 2021 +0100
```

Obsérvese que el proyecto consta únicamente de dos commits por el momento. El primer commit (ID: 48fe2767…) fue el que realizó el autor del repositorio original "hello-world" en diciembre de 2019. Es decir, es la primera versión del repositorio. El segundo commit (ID:00056aa4…) ha sido nuestra segunda versión, en la cual hemos descrito nuestros cambios como "Trasteando" y donde hemos incorporado el nuevo archivo de "modificacion.txt".

Git push

Con el anterior comando **git commit** actualizamos los cambios en nuestra copia local local, pero también deseamos que estos cambios se reflejen en el repositorio remoto. Para ello emplearemos el comando **git push**, el cual actualizará las modificaciones en el repositorio original.



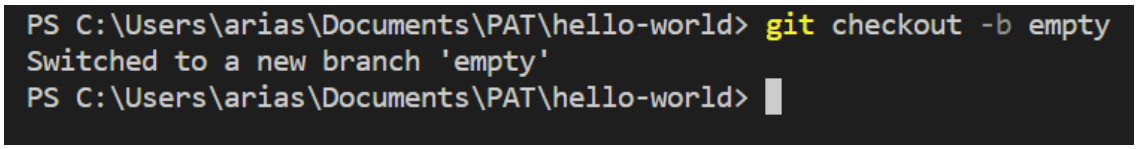
En la figura se resalta en rojo el cambio en el ID del commit. Es decir, se ha pasado de la primera versión de proyecto (ID: 48fe276…) a la segunda (ID:00056aa…).

En esta ocasión no ha sido necesario ejecutar **git push origin <nombreRama>** porque contamos únicamente con la rama "main". Si hubiéramos tenido más ramas en el repositorio, tendríamos que haber especificado sobre cuál desearíamos ejecutar **git push**.

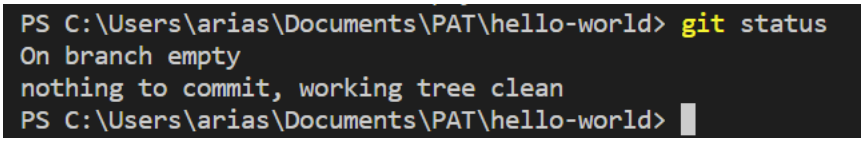
Git checkout

El comando **git checkout** nos permite trabajar en distintas ramas de nuestro proyecto. Además, este comando también nos permite volver a versiones anteriores de nuestro repositorio empleando el ID del commit al que nos queramos remontar.

Este proyecto se caracteriza por tener una única rama denominada "main". Para probar a navegar entre diferentes ramas, creamos una rama adicional llamada "empty".

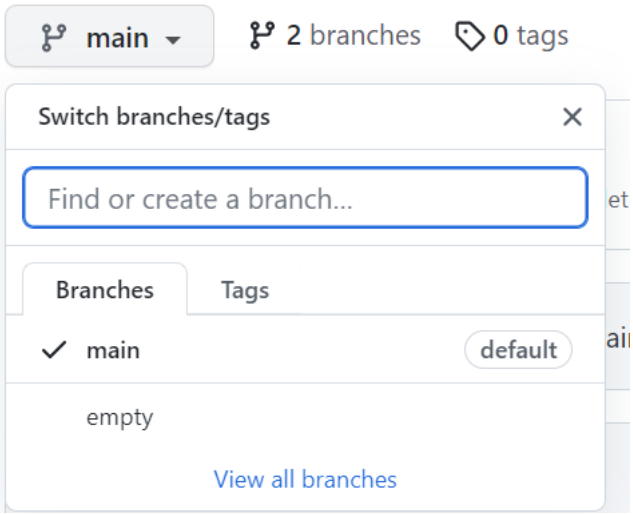


Confirmamos que hemos creado esta nueva rama y que nos encontramos en ella con **git status**.



Sin embargo, como todavía no hemos actualizado este cambio en nuestro repositorio, no tenemos constancia de la existencia de esta nueva rama:

Para ello habrá que ejecutar **git push** de manera que se actualice la estructura del proyecto. Una vez ejecutado, ya aparece la rama "empty" en nuestro repositorio.



Si queremos volver a trabajar en la rama principal “main”, basta con ejecutar el comando **git checkout main** en el terminal.

```
PS C:\Users\arias\Documents\PAT\hello-world> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\arias\Documents\PAT\hello-world> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
PS C:\Users\arias\Documents\PAT\hello-world>
```

Como ya hemos terminado de probar estos comandos para movernos por diferentes ramas, eliminaremos la rama “empty” para dejar el repositorio como estaba antes de esta modificación. La eliminación de una rama se puede hacer con el comando **git branch -d <nombreRama>.**

```
PS C:\Users\arias\Documents\PAT\hello-world> git branch -d empty
Deleted branch empty (was 00056aa).
PS C:\Users\arias\Documents\PAT\hello-world>
```

Prueba de instalación

Como requisito de esta práctica también se requería la instalación de software.

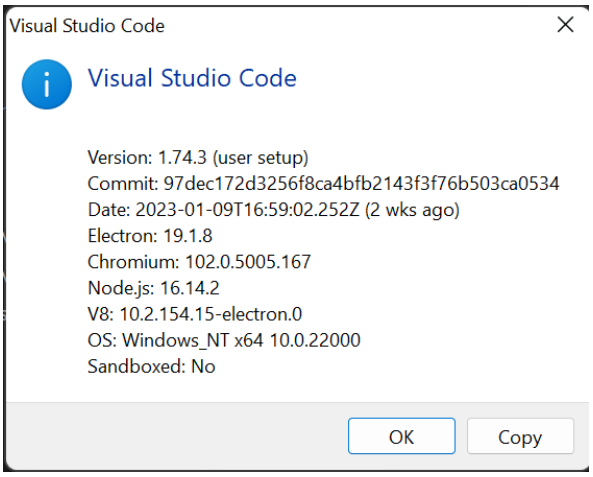
Java 17

```
C:\Users\arias>java -version
java version "17.0.6" 2023-01-17 LTS
Java(TM) SE Runtime Environment (build 17.0.6+9-LTS-190)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.6+9-LTS-190, mixed mode, sharing)

C:\Users\arias>
```

Maven

```
C:\Users\arias>mvn -version
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: C:\maven
Java version: 17.0.6, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Users\arias>
```

Editor de código fuente: VSCode



Docker

```
C:\Users\arias>docker version
Client:
 Cloud integration: v1.0.29
 Version:           20.10.22
 API version:       1.41
 Go version:        go1.18.9
 Git commit:        3a2c30b
 Built:             Thu Dec 15 22:36:18 2022
 OS/Arch:           windows/amd64
 Context:           default
 Experimental:      true

Server: Docker Desktop 4.16.2 (95914)
 Engine:
  Version:          20.10.22
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.18.9
  Git commit:       42c8b31
  Built:            Thu Dec 15 22:26:14 2022
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.14
  GitCommit:        9ba4b250366a5ddde94bb7c9d1def331423aa323
 runc:
  Version:          1.1.4
  GitCommit:        v1.1.4-0-g5fd4c4d
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```