Juan Pardo de Santayana Navarro
Programación de Aplicaciones Telemáticas – 3ºB GITT-BA

Práctica 1 – Entorno de Desarrollo – Comandos Git

En esta práctica se van a probar distintos comandos básicos de GitHub y se documentará su utilidad y sus resultados.

1- git clone

```
@jpardods ➜/workspaces/p1-fork (main) $ git clone https://github.com/gitt-3-pat/p1
Cloning into 'p1'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 5
Receiving objects: 100% (6/6), done.
```

Este comando sirve para realizar una copia de un repositorio completo existente en un nuevo directorio. Una vez se ha realizado se ha clonado el repositorio, se habrá guardado una copia del mismo en la máquina local. De esta manera, se puede trabajar sobre ese código y realizar cambios, para más adelante actualizar estos cambios utilizando commits y push al repositorio propio o al repositorio original.

En este caso, se ha utilizado para copiar el repositorio de la práctica 1, es decir, *https://github.com/gitt-3-pat/p1*. Gracias a este comando, se pueden realizar cambios, pruebas y experimentos sobre la práctica sin que estos afecten al repositorio original.

Fijándose ahora en los logs que deja el comando, se puede observar que lo primero que hace es especificar en el directorio en el que se copia el repositorio: “Cloning into ‘p1’”, que significa que el repositorio se está clonando dentro del directorio p1. El siguiente log enumera el número de objetos encontrados en el repositorio, que son 6. Acto seguido se cuentan el total de objetos, de los cuáles 5 fueron empacados y reutilizados, un método de GitHub para empaquetar múltiples objetos en un solo archivo, lo que reduce el tamaño de la transferencia de datos. El último log indica que se han recibido todos los objetos del repositorio original.

2- git status

```
@jpardods ➜/workspaces/p1-fork (main) $ 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)
        p1/

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

Este segundo comando proporciona información sobre el estado actual del repositorio local en relación con el repositorio remoto que ha sido clonado. Este comando puede tener distintas respuestas posibles que ofrecen detalles sobre los cambios y actualizaciones de estos repositorios. Las respuestas posibles son las siguientes: “Untracked files”, que muestra la lista de archivos que no están siendo rastreados por el GitHub ya que no se han añadido todavía al área de preparación; “Changes to be committed”, que son cambios listos en el área de preparación para ser cometidos; “Changes not staged for commit”, que son cambios que se han realizado en el directorio de trabajo pero no se han incluido en el área de preparación todavía; y “On branch” que indica la rama en la que se encuentran en el momento. Además, puede ofrecer información adicional, como ramas remotas o detalles relevantes.

En este caso, se ve que se encuentra en la rama *main* ("On branch main"), ques esta actualizada con el *origin/main*. En la parte de "Untracked files", se encuentra el repositorio que había sido clonado en el comando anterior (*/p1*), ya que todavía no ha sido añadido al directorio de trabajo ni se ha incluido en el área de preparación. Esta información se obtiene de los logs que devuelve el comando git status.

3- git add .

```
@jpardods ➜/workspaces/p1-fork (main) $ git add .
warning: adding embedded git repository: p1
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> p1
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached p1
hint:
hint: See "git help submodule" for more information.
```

Este tercer comando se utiliza para agregar todos los archivos que hayan sido añadidos o modificados en el directorio de trabajo al área de preparación. El área de preparación (*staging* área) es el espacio intermedio entre el directorio de trabajo y el repositorio Git, en el que se colocan todos aquellos archivos que se quieren incluir en el siguiente commit. El punto tras el add significa que se quieren añadir todos los archivos del directorio de trabajo en el área de preparación.

En este caso, se observa una advertencia de que se que se está añadiendo el repositorio incorporado *p1* al área de trabajo. El primer log indica que se está añadiendo un nuevo repositorio dentro del repositorio actual, que es lo que se busca. Además, el resultado del comando ofrece distintos mensajes de ayuda y de advertencia, que no son aplicables en esta práctica, como que los clones del repositorio externo que se está añadiendo no se incluyen en el repositorio incorporado.

4- git commit -m "añadir repositorio p1"

```
@jpardods ➜/workspaces/p1-fork (main) $ git commit -m "añadir repositorio p1"
[main 1142f60] añadir repositorio p1
 1 file changed, 1 insertion(+)
 create mode 160000 p1
```

Este comando se utiliza para realizar un commit en el repositorio de GitHub con un mensaje asociado que sirve de confirmación. Sirve para capturar y guardar todos aquellos cambios realizados en el directorio de trabajo y que previamente han sido añadidos al área de preparación mediante el comando *git add*. La parte del *git commit* es la que crea el nuevo commit y la parte de *-m "añadir repositorio p1"* sirve para asociar este mensaje de confirmación al commit y, en general, proporciona una descripción de los cambios que se están realizando en el repositorio. Un *commit* es como si fuese una captura del repositorio, ya que

guarda todos los cambios realizados y preparados con una breve descripción en un momento determinado en el repositorio local, creando una nueva versión de ese proyecto.

En este caso, el primer log que se recibe es una confirmación de que el *commit* se ha realizado en el *main,* indica el número indentificador del *commit* y enseña el mensaje asociado al *commit*. El siguiente log indica el número de archivos que han sido modificados o insertados, que en este momento es tan solo 1. El último log indica que se ha creado un nuevo modo para el directorio llamado *p1*.

5- git push

```
@jpardods →/workspaces/p1-fork (main) $ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 286 bytes | 286.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
remote: This repository moved. Please use the new location:
remote:   https://github.com/jpardods/p1.git
To https://github.com/jpardods/p1-fork
   07720b5..1142f60  main -> main
```

Este quinto comando sirve para enviar los *commits* realizados en el repositorio local a un repositorio remoto. De esta manera se consigue actualizar el repositorio remoto que ha sido clonado con los cambios que se han realizado en la rama local.

Lo primero que hace este comando es enumerar y contar los objetos locales que se envían al repositorio remoto. Gracias al siguiente log, la compresión delta permite minimizar la cantidad de datos que se envían al repositorio remoto. De misma manera, el log “Compressing objects” indica que se están comprimiendo los objetos que se deben enviar antes de enviarse. Acto seguido, se escriben los objetos enviados en el repositorio remoto y se proporciona información sobre la velocidad de este proceso. Tras indicar que los 2 objetos totales han sido recibidos, Git indica que el repositorio en el que se encontraba se ha movido y proporciona la información del movimiento: la nueva ubicación del repositorio es *https://github.com/jpardods/p1.git*, desde el que se ha realizado el *push* al repositorio remoto *https://github.com/jpardods/p1-fork*. Por último, el comando realiza un resumen de los cambios enviados: 07720b5 es el indicador del *commit* anterior al *push* y 1142f60 es el nuevo indicador tras el *commit* en la rama *main*.

6- git checkout -b feature/1

```
@jpardods →/workspaces/p1-fork (main) $ git checkout -b feature/1
Switched to a new branch 'feature/1'
```

El comando *git checkout* se utiliza para moverse de una rama a otra. El hecho de añadir *-b* al comando sirve para que, en caso de que la rama indicada no exista, esta rama se cree y además se mueva a ella. Normalmente, las ramas nuevas se crean para trabajar en una parte específica del proyecto sin afectar la rama definitiva *main*.

En este caso, se ha creado una nueva rama llamada *feature/1* y se ha movido hasta ella. En el log que aparece se notifica de dicho movimiento.

7- git checkout main

```
@jpardods →/workspaces/p1-fork (feature/1) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Este último comando de la primera práctica se utiliza para moverse de la rama actual a la rama "main" en el repositorio de GitHub. Con esto, se encontrará en la rama "main" y el directorio de trabajo se actualizará con los archivos y el estado de la rama principal.

El primer log indica el movimiento a la rama "main" y el segundo log confirma que se encuentra actualizada con la rama del repositorio local *origin/main*.

El resultado de todos estos comandos en el repositorio local es el siguiente: