Carlos Sánchez-Cabezudo Pinto 26/01/2024
Programación de Aplicaciones Telemáticas

Práctica 1 – Entorno de desarrollo

Comandos utilizados:

- Git clone: Se hace uso de este comando de la siguiente forma, teniendo en cuenta el nombre del usuario y el repositorio:

```
@carlossanchezcabezudo ➜/workspaces/p1-fork (main) $ git clone https://github.com/carlossanchezcabezudo/p1-fork
Cloning into 'p1-fork'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 12 (delta 2), reused 2 (delta 2), pack-reused 9
Receiving objects: 100% (12/12), done.
Resolving deltas: 100% (3/3), done.
```

El comando utilizado se utiliza para clonar un repositorio git. A continuación de este comando se incluye la URL del repositorio que se está clonando. En este caso, como se puede observar en la foto superior, se está clonando el repositorio ubicado en GitHub del usuario "carlossanchezcabezudo" con el nombre "p1-fork".

En las líneas que siguen al comando se llevan a cabo las siguientes acciones:

Ennumerating objects: Git se comunica con el servidor para obtener información de los objetos que conforman al repositorio.

Counting objects: se cuenta el número de objetos que serán transferidos.

Total y reused: esta línea indica el total de objetos a transferir y cuántos de estos se están reutilizando o transfiriendo nuevamente por ya existir localmente.

Receiving objects: Git descarga los objetos del repositorio remoto a la máquina local.

Resolving deltas: se resuelven los "deltas", es decir, los cambios entre versiones para que la información se ajuste correctamente en la copia local.

Una vez finalizado este proceso, se obtiene una copia completa del repositorio "p1-fork" en la máquina local, y se ubica en el directorio "p1-fork" en el terminal de comandos (main).

- Git status:

```
@carlossanchezcabezudo ➜/workspaces/p1-fork (main) $ 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)
        modified:   README.md
        new file:   git.txt

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    git.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html
        p1-fork/
```

Este comando se utiliza con el fin de obtener información sobre el estado actual del repositorio. Se obtiene el output mostrado en la imagen superior del que cabe explicar distintos aspectos:

En primer lugar, on branch main indica que actualmente se está situado en la rama “main”.

La siguiente línea indica que la roma local 2main” está actualizada con respecto a la rama remota “main2 en el repositorio remoto llamado “origin”. Este repositorio remoto “Origin” suele hacer referencia al repositorio remoto desde el cual se ha clonado el repositorio local.

La línea “Changes to be committed” muestra los cambios que están listos para ser confirmados en el próximo commit. Los que se pueden observar son los siguientes: “modified: README.md” ya que se han realizado cambios en este archivo y “new file: git.txt” ya que se ha creado un nuevo archivo con este nombre.

Los “Changes not staged to commit” son aquellos cambios que han sido realizados, pero aún no han sido incluidos en el área de preparación (es decir, el staging área) para el próximo commit. Como se puede observar en la imagen, el cambio observado es la eliminación del archivo “git.txt”.

Por último, la lista de “Untracked files” muestra archivos que existen en tu directorio de trabajo pero que no están siendo por el Git. En el caso que nos ocupa estos son: “index.html” y “p1-fork”.

Solucionando los avisos previos, se consigue que quede el siguiente resultado:

```
@carlossanchezcabezudo →/workspaces/p1-fork (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

La información proporcionada por Git status es útil para entender qué cambios se han realizado, qué cambios están listos para ser confirmados y qué archivos o directorios aún no están bajo el control de Git.

- Git add:

```
@carlossanchezcabezudo →/workspaces/p1-fork (main)
$ git add .
```

Este comando se utiliza para agregar cambios al área de preparación (es decir, el staging área) antes de realizar un commit posterior. Es decir, permite seleccionar los cambios a incluir en el próximo commit (estos pueden ser nuevos archivos, archivos modificiados o eliminación de archivos del área de preparación).

Como a continuación del comando add se incluye un punto, se incluyen todos los archivos y cambios. Se muestra el siguiente mensaje:

```
@carlossanchezcabezudo →/workspaces/p1-fork (main)
$ git add .
warning: adding embedded git repository: p1-fork
hint: You've added another git repository inside you
r current repository.
hint: Clones of the outer repository will not contai
n 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-fork
hint:
hint: If you added this path by mistake, you can rem
ove it from the
hint: index with:
hint:
hint:   git rm --cached p1-fork
hint:
hint: See "git help submodule" for more information.
```

Los mensajes de advertencia indican que se ha intentado agregar al área de preparación “git add” un directorio “p1-fork” que ya contiene un repositorio Git dentro del repositorio actual. Esto es algo potencialmente problemático, pues agregar un repositorio Git dentro de otro (sin usar submódulos) puede llevar a problemas de seguimiento y gestión.

Se incluyen sugerencias de como atajar este problema:

Si se pretendiese añadir un submódulo se indica que se puede utilizar el siguiente código: git submodule add <url> p1-fork, que configurará “p1-fork” como un submódulo vinculado al repositorio principal.

Si se ha agregado “p1-fork” por error se puede eliminar la referencia al directorio p1-fork del área de preparación con el siguiente comando:

git rm --cached p1-fork. Sin embargo, el directorio en sí seguirá existiendo en el sistema de archivos del usuario.

- Git commit: Se hace uso de este comando añadiendo a continuación el mensaje en cuestión:

```
@carlossanchezcabezudo →/workspaces/p1-fork (main) $ git commit -m "SE COPIA EL REPOSITORIO"
[main 8e22e88] SE COPIA EL REPOSITORIO
 1 file changed, 1 insertion(+)
 create mode 160000 p1-fork
```

Este comando se utiliza para confirmar los cambios realizados en el área de preparación y crear un nuevo commit en la historia del repositorio. Además, cabe destacar que el comando commit representa un conjunto de cambios atómicos y viene acompañado de un mensaje descriptivo que explica el propósito de estos cambios. En este caso, tal y como se muestra en la imagen superior, el mensaje es “SE COPIA EL REPOSITORIO” (se utiliza -m antes del mensaje para proporcionar el mensaje de commit directamente desde la línea de comandos).

La salida muestra información acerca del resultado del commit:

“[main 8e22e88] SE COPIA EL REPOSITORIO”: esto indica que se ha creado un nuevo commit en la rama "main" con el identificador 8e22e88 y el mensaje en cuestión.

"1 file changed, 1 insertion(+)": informa que se ha realizado un cambio en un archivo y se ha insertado una línea.

"Create mode 160000 p1-fork": muestra que se ha creado un nuevo estado en el área de preparación con respecto al subdirectorio p1-fork. En concreto en este caso se usa el estado 16000, que suele estar asociado con submódulos en Git.

Cabe deducir de la última línea explicada que el directorio "p1-fork" ha sido marcado como un submódulo (tiene sentido considerando las advertencias ya explicadas que aparecían tras la ejecución del comando add). De esta forma, se pueden gestionar los repositorios de forma separada pero vinculada.

- Git push:

```
@carlossanchezcabezudo →/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), 347 bytes | 347.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/carlossanchezcabezudo/p1-fork
   c84068a..8e22e88  main -> main
```

El comando "git push" se utiliza para enviar los commits locales a un repositorio remoto actualizando en este proceso la rama correspondiente. Se ha de mencionar también lo que implica las salidas obtenidas usando este comando:

"Enumerating objects": git cuenta los objetos que se están enviando al repositorio remoto (en esta caso 3).

"Counting objects": indica que se han contado todos los objetos mencionados previamente (se puede apreciar 3 de 3 contados).

"Delta compression using up to 2 threads": se utiliza la comprensión delta para reducir el tamaño de los datos que se están enviando.

"Compressing objects": muestra que la comprensión ha finalizado.

La penúltima línea informa sobre el número total deobjetos enviados y cómo se están manejando las deltas (es decir, las diferencias entre las distintas versiones).

Por último, se muestra el repositorio remoto al que se están enviando los cambios.

- Git checkout:

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

El comando git checkout se utiliza para cambiar de ramas o crear una nueva rama. En este caso, se utiliza para crear una nueva rama y luego cambiar entre la rama recién creada y la principal (“main”).

El primer comando se utiliza para crear una nueva rama y cambiar a ella al mismo tiempo. Esta rama recibe el nombre de “feature/1”.

El segundo comando se utiliza para cambiar a la rama “main”. La aclaración "Your branch is up to date with 'origin/main'" significa que la rama local "main" está sincronizada con la rama remota "main", lo que indica que no hay cambios nuevos en la rama remota que no se hayan descargado.

Este proceso de cambiar a una rama nueva y luego volver a la rama “main” es habitual cuando se trabaja con ciertas características como la corrección de errores en ramas separadas antes de fusionar esos cambios en la rama principal.