Terminal, git, GitHub y GitHub pages

Parte II

{desafío} latam_

- Objetivo de la sesión
- Conceptualización
- Ejercitación
- Reflexión

Inicio

10 minutos

{desafío} latam_

Objetivos de la sesión

/*Aplicar el procedimiento de subida del código versionado, mediante una conexión con github desde la terminal para la mantención de un repositorio remoto.*/

/*Aplicar el procedimiento de habilitación de una página web con Github Pages, para que el contenido esté disponible públicamente.*/

Desarrollo

90 minutos

{desafío} latam_

Existen varios tipos de repositorios remotos y empresas asociadas a proporcionarlos, las más usadas son: GitHub, Bitbucket y Gitlab.

Existen varias formas de hacerlo, pero la más fácil y segura es utilizar el protocolo SSH y sus llaves. Las llaves, son un medio por el cual podemos identificar nuestro equipo con un servidor o página específica, incluso sin tener que ingresar una contraseña. Esto funciona mediante dos llaves, una privada y otra pública. La privada vive en nuestro equipo y la pública es la que ingresa en el lado remoto.

Subiendo y bajando cambios

Subiendo y bajando cambios

Para subir los cambios al repositorio remoto debemos utilizar el comando:

```
git push origin main
```

Subiendo y bajando cambios

Ahora, si por el contrario necesito bajar los cambios que están en un remoto que tenemos registrado en nuestro proyecto, podremos utilizar el comando:

```
git pull origin main
```

Subiendo y bajando cambios

Manejo de repositorios remotos

Si necesitamos saber si el proyecto en que estamos trabajando ya contiene alguna referencia a un repositorio remoto, lo realizaremos con el comando:

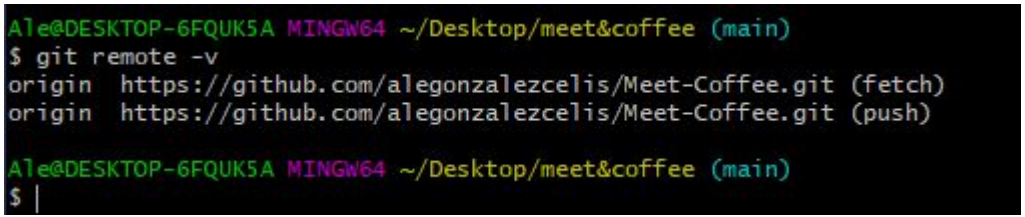
```
git remote
```

Subiendo y bajando cambios

Manejo de repositorios remotos

Si necesitamos saber las url de estos servidores usamos el comando:

```
git remote -v
```



Subiendo y bajando cambios

Añadiendo un repositorio remoto

Para añadir un repositorio remoto, simplemente debemos usar el comando:

```
git remote add [nombre] [dirección del repositorio]
```

Subiendo y bajando cambios

Obteniendo información de un repositorio remoto

Podremos obtener información de él con el siguiente comando:

```
git remote show [nombre]
```

Subiendo y bajando cambios

Manejo de repositorios remotos

Si necesitamos renombrar un repositorio remoto que hemos añadido, podemos realizarlo de la siguiente forma:

```
git remote rename nombreActual  NuevoNombre
```

Subiendo y bajando cambios

Manejo de repositorios remotos

Si necesitamos borrar un repositorio remoto, podemos realizarlo con el siguiente comando git:

```
git remote rm NombreRepositorio
```

Subiendo y bajando cambios

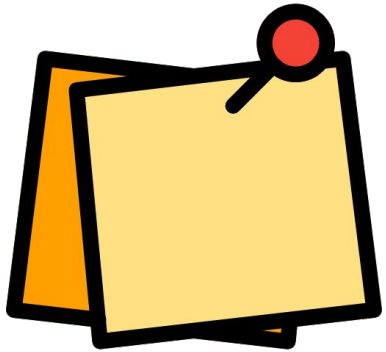
Repositorios que ya están en GitHub

GitHub nos proporciona herramientas para utilizar repositorios que ya están creados. Para ellos existen dos herramientas que nos ayudarán: Fork y Clone.

```
git clone [dirección del repositorio]
```

Subiendo y bajando cambios

Importante



GitHub en sus inicios definió la rama master como su rama principal o por defecto, sin embargo, desde octubre de 2020, han implementado cambios en su rama principal, sustituyendo gradualmente la rama master por la nueva rama main, este cambio afecta directamente a los nuevos repositorios. Según se puede apreciar en su documentación oficial, el cambio responde a la implementación de un nombre más corto y a la interpretación más óptima en otros lenguajes. Para más información acerca de esta actualización visita el siguiente link: https://github.com/github/renaming

Como vimos en el capítulo anterior, Git trabaja a modo de repositorio local y al finalizar cada cambio, podremos guardar los avances en el repositorio remoto de Github.

Para esto, ejercitaremos creando un repositorio remoto para el proyecto, lo añadiremos por consola y subiremos sus cambios.

Ejercicio guiado: Utilizando GitHub

Git branch

Git branch

Conociendo las ramas del proyecto

Una de las principales ventajas de git es la utilización de branch (ramas). Primero definiremos una rama simplemente como un camino donde está nuestro código, mientras que la definición técnica se refiere a un branch como un apuntador a donde van los commits que realizamos.

Cada vez que inicializamos git, de forma automática se genera un branch main, que es donde se guardarán los nuevos commits.

Git branch

Conociendo las ramas del proyecto

Al iniciar git en una carpeta, de forma automática se genera la rama main.
Podremos conocer las ramas que tiene nuestro proyecto con el comando:

```
git branch
```

Git branch

¿Cuándo generar una nueva rama?

Para crear una nueva branch utilizaremos el siguiente comando:

```
git branch nueva_branch
```

Git branch

¿Cuándo generar una nueva rama?

Una vez creado, no cambiará de forma automática a la nueva rama, sino que nos quedaremos en la rama original. Para cambiarnos debemos hacerlo con el comando:

```
git checkout nueva_branch
```

Git branch

¿Cuándo generar una nueva rama?

Si queremos crear una branch y situarnos enseguida en ella debemos ejecutar el siguiente comando:

```
git checkout -b otra_branch
```

Git stash

Git stash

Existe una manera de reservar o “apartar” las modificaciones que estamos haciendo en nuestra rama actual para realizar una tarea emergente y asegurarnos que no perderemos los cambios que hemos realizado, para esto podemos optar por ocupar el comando:

```
git stash
```

Git stash

todos nuestros cambios han desaparecido, sin embargo, lo que sucedió es que fueron diferidos a un área en paralelo de nuestro historial de commits al cual podemos acceder con el siguiente comando:

```
git stash list
```

Git stash

Para volver a unir el directorio actual con los cambios que estábamos trabajando y que fueron apartados del commit en el que nos encontrábamos.

```
git stash apply
```

Git Rebase

Git rebase

Cuando estamos trabajando en un equipo de desarrollo y varios usuarios manipulan el mismo repositorio en paralelo, es normal que se generen muchas ramas y muchos commits que dificultan la lectura del historial general, puesto que habrán modificaciones agrupadas en una rama A que serán unidas a una rama B a partir del último cambio generado.

Para lograr un historial más limpio con menos ramas y menos commits distribuidos, git nos ofrece el siguiente comando:

```
git rebase otra_branch
```

GitHub Pages

GitHub Pages

¿Qué es GitHub pages?

GitHub pages y renderiza nuestro código de una rama específica en un dominio y espacio dentro de GitHub.

GitHub Pages

Subiendo una página a GitHub pages

Hay dos formas de realizar la subida a GitHub pages. La primera es manual a través de la consola:

```
git branch gh-pages
```

GitHub Pages

Subiendo una página a GitHub pages

Ahora, pasaremos todos los commits de la rama main a la nueva rama, para luego subir los cambios al repositorio remoto.

```
git checkout gh-pages
```

```
git merge main
```

```
git push origin gh-pages
```

GitHub Pages

Subiendo una página a GitHub pages

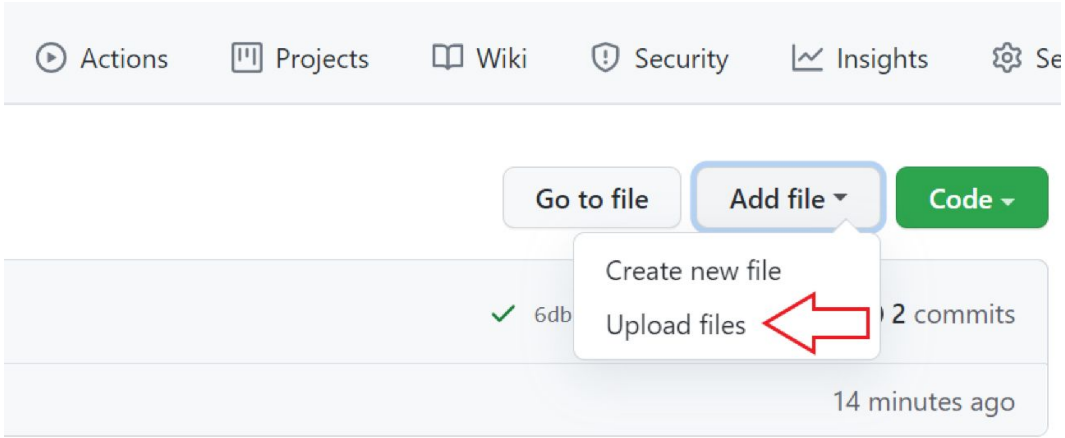
Finalmente, visitamos la página (reemplaza con tu nombre de usuario y nombre de tu repositorio donde corresponde).

```
https://TuNombreDeUsuario.GitHub.io/Nombre_de_tu_repositorio/
```

GitHub Pages

Subiendo una página a GitHub pages

La segunda forma de hacer un despliegue de sitios web en este servicio es a través de la interfaz gráfica de Github, entrando al repositorio encontraremos un botón desplegable para agregar archivos a una rama y posteriormente arrastrar los mismos desde de el explorador del sistema operativo, así como se muestra en las siguientes imágenes.



{desafío} latam_

Vamos a utilizar git dentro de un proyecto. En nuestro caso, vamos a iniciar git en el sitio creado con HTML y CSS de un proyecto anterior llamado "meet&coffee".

Ejercicio guiado: Utilizando git en un proyecto (Parte I)

{desafío} latam_

Quizzes

15 minutos

{desafío} latam_

Cierre

15 minutos

{desafío} latam_

Reflexionemos

{desafío}
latam_

{desafío} latam_

Academia de talentos digitales