Contenido

No se encontraron entradas de tabla de contenido.

Referencias

Video de EDTeam desde 0 los conceptos básicos

https://www.youtube.com/watch?v=jGehuhFhtnE

Video del canal bluuweb

https://www.youtube.com/watch?v=hWglK8nWh60

Web de bluuweb con los conceptos básicos

https://bluuweb.github.io/desarrollo-web-bluuweb/07-01-git/#enlaces

Video canal Bye Code. Ejemplo básico y ejemplo con gitflow

https://www.youtube.com/watch?v=XpulbHt5zZ0&t=2510s

Dibujo en blanco y negro

Descripción generada automáticamente con confianza bajaholaMundo

https://www.youtube.com/watch?v=VdGzPZ31ts8

------------------

Conceptos

Una captura de un videojuego

Descripción generada automáticamente con confianza baja



Tratando de explicar la imagen: Tenemos nuestro directorio local (una carpeta en nuestro pc) con muchos archivos, Git nos irá registrando los cambios de archivos o códigos cuando nosotros le indiquemos, así podremos viajar en el tiempo retrocediendo cambios o restaurando versiones de código, ya sea en Local o de forma Remota (servidor externo). En la práctica quedará más claro.

Git Comandos básicos en la terminal de Git Bash en Windows

Si estás empezando a utilizar el sistema de control de versiones de Git, estos comandos te serán útiles para navegar y hacer operaciones entre tus carpetas y archivos en Windows.

Primero debemos recordar la diferencia en la estructura de archivos de Windows, Mac o Linux.

La ruta principal en Windows es c:/, en UNIX es solo /.

Windows no hace diferencia entre mayúsculas y minúsculas, pero UNIX si.

Git Bash usa la ruta /c para dirigirse al disco c:/ en Windows. Por lo tanto, la ruta del usuario con el que estás trabajando es */c/Users/NombreUsuario*

Comandos

pwd: nos muestra la carpeta actual en la que nos encontramos.

mkdir: nos permite crear carpetas, p. ej. mkdir *NuevaCarpeta*

touch: nos permite crear archivos nuevos, p.ej. touch *NuevoArchivo.txt*

cat: nos permite ver el contenido de un archivo, p.ej. cat *NuevoArchivo.txt*

cd: nos permite cambiarnos de carpeta, p.ej. cd *NuevaCarpeta.*

cd .. : nos permite regresar al directorio o carpeta anterior.

cd o cd ~: nos lleva a la ruta del usuario.

cd /c: nos vamos al disco C:/.

cd -: nos lleva directamente al ultimo directorio visitado.

ls: nos permite ver los archivos de la carpeta donde estamos actualmente.

ls -l: Ver todos los archivos como una lista en donde incluye el usuario, grupo, permisos sobre el archivo, tamaño, fecha y hora de creación.

ls -lh: Muestra la misma información que ls-l pero con unidades de tamaño, es decir, kb o mb.

ls-R: muestra el contenido de todos los sudirectorios de forma recursiva.

ls -S Ordena los resultados por tamaño de archivo.

rm: Nos permite borrar un archivo o carpeta ej: rm NuevoArchivo.txt

rmdir “nombre del directorio”/: borrar un directorio: Solo funciona con directorios vacíos.

rm -r ‘nombre de la carpeta’ :me permite eliminar la carpeta y los archivos dentro de ella de forma recursiva.

cp “nombre del archivo que quremos copiar” “nombre del directorio a donde lo queremos copiar”: nos permite copiar un archivo.

mv “el directorio de donde queremos mover/el nombre del archivo” “el directorio hacia donde lo queremos mover”: nos permite mover un archivo.

clear: nos permite limpiar la pantalla.

history: ver los últimos comandos que ejecutamos y un número especial con el que podemos volver a repetir el comando.

Primeros pasos

¡Importante!

Utilizar la consola de git bash no la cmd de Windows ni power Shell

//Versión de git (sirve para comprobar que lo tenemos bien instalado)

git version

//Registro de usuario asociado a git. La opción global es para que se configure para todos los proyectos frente a local, para el proyecto actual.

git config --global user.name "mi nombre"

git config --global user.mail "mi correo"

(muy conveniente utilizar el nombre y correo de github).

Para ver estos datos

git config user.name

git config user.mail

//para señalar el editor de texto VSCode como editor por defecto. –wait hace que el terminal espera a que se cierre el editor

git config --global core.editor “code --wait”

//para ver el fichero de configuración global, se abre en vscode

git config --global -e

// comportamiento del CRLF true si Windows input si mac o linux

git config --global core.autocrlf true

// Importante: de cara a

git config --global http.proxy https://proxy.jcyl.es:80

//Para ver los parámetros de configuración de git

git config --list

//Mi primer repositorio (sólo una vez por proyecto)

git init

// para ver el estado de los archivos (si han sido registrados o no)

git status

git status -s //versión resumida

//Añadir a la staged área

git add . // todos los ficheros

git add \*.html

//Confirmar los cambios (hacer una todo del proyecto, enviarlo al repositorio)

git commit -m “Comentario con sentido”

Es obligatorio un comentario si no le escribimos el parámetro -m nos abrirá el editor por defecto