RESUMEN GIT

Instalación

**sudo apt-get update**

**sudo apt-get install git**

Configuracion

git config --global user.name "Your Name"

git config --global user.email "youremail@domain.com"

Opcionalmente el editor (si no me gusta el que hay por defecto):

git config --global core.editor vi

Para todos los usuarios: /etc/gitconfig

Para un usuario: ~/.gitconfig (opción --global)

Para un repositorio: .git/config

git config –list

SSH

ssh-keygen

Copiaremos el contenido de ~/.ssh/id\\_rsa.pub a una nueva clave ssh en GitHub

echo "# GIT\_SERVIDOR" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin git@github.com:JavierMonforte/GIT\_SERVIDOR.git

git push -u origin main

**Me muevo al directorio-**

Commit inicial

git init

echo "# Repositorio de ejemplo" >> README.md

git add README.md

git commit -m "Commit inicial"

**Creo repositorio en github**

git branch -M main

git remote add origin git@github.com:JavierMonforte/GIT\_SERVIDOR.git

git push -u origin main

**Si parto de mi repositorio**

git clone <url repo> #habitual

git clone <url repo> <carpeta> #permite definir destino

**Guardar cambios**

Consultar el estado del repositorio: git status

git add .

git commit -m "comentario explicativo"

git add <nombre fichero> #un sólo fichero

git add img/logo.jpg

git add <nombre directorio> #todo lo de un directorio

git push

**Marcha Atrás De Un Fichero Modificado**

git checkout <nombrefichero>

git checkout casa.md

git checkout <nombre directorio>

git checkout .

**Queremos sacar del estado preparado después de git add:**

git reset HEAD <nombrefichero>

git reset HEAD casa.md

git reset HEAD <nombre directorio>

git reset HEAD .

git push -u origin master

git fetch

git status

git pull

**Al acabar la sesión de trabajo**

git commit

git status

git push

**RAMAS (BRANCH)**

git branch // lista de ramas

git branch <rama> // crear rama

git checkout <rama> // cambiar de rama

git checkout -b <rama> // crear y cambiar rama 2 en 1

git branch -d <rama> // borrar rama

**FUNDIR RAMAS**

git checkout master // nos ponemos en rama master

git merge <rama> //fundimos con la rama deseada

**GIT PUSH**

git push # sube rama "preferida"

git push <repo remoto> <rama> # sube una rama concreta

git push origin dev # Ej. sube rama dev

git push -u <repo remoto> <rama># sube y predetermina rama

git push -u origin master # Ej. sube y pred. Master

**RAMAS REMOTAS**

git push origin <rama>

Ambas crear la rama local asociada a origin/

La primera nos permite alterar el nombre .

git checkout -b <rama> origin/<rama>

git checkout --track origin/<rama>

**REVISANDO CÓDIGO**

git log nos muestra información histórica

git diff compara el contenido de los ficheros

**GIT LOG**

git log #uso base

git log -<n> #log de los últimos n commits

git log --oneline -5 #lista de commits breve

git log --follow [file] #lista de commit con cambios para "file"

git show <commit> # información completa de un commit concreto

**GIT DIFF**

git diff # diferencia entre estado actual y el preparado o comprometido

git diff --cached #diferencia entre el preparado y el último commit

git diff --stat #idem al primero pero con información resumida de cambios

**DIFF DESDE UN COMMIT CONCRETO**

git diff <commit> # cambios desde un commit hasta la actualidad

git diff <commit> <file># idem fichero concreto

git diff master~2 README.md #Cambios en los dos últimos commit de master

git diff HEAD~2 README.md #Cambios en los dos últimos commit HEAD

git diff cd598e4 README.md #Cambios desde un commit concreto por hash

**DIFF ENTRE DOS COMMIT**

git diff <commit>:<file> <commit2>:<file> #diff de un fichero entre dos commits

git diff cd598e4:README.md 31422ac:README.md

git diff master~20:README.md master~1:README.md

**GITIGNORANDO COSAS: .GITIGNORE**

MARKDOWN

✗ Párrafos. Dos veces intro. Pone una línea en blanco.

✗ Saltos de línea. No termina el párrafo, pero continúa escribiendo en la siguiente línea. Dos espacios más intro.

✗ Encabezados. Usar una almohadilla, #, por número de encabezado hasta 6. No hace falta cerrar el encabezado.

▪ Otra forma de hacer encabezados 1 y2 es poniendo debajo de la línea del encabezado un =, igual, para h1 y –, guion, para h2.

✗ Citas. Se añade un >, mayor que, al inicio del bloque.

▪ Si la cita consta de varios párrafos hay que poner un > antes de cada párrafo. ▪ Se pueden concatenar varios >> para hacer citas anidadas.

✗ Listas.

▪ Desordenadas. Se usa delante de cada elemento de la lista un \*, asterisco, -, guion o +, suma. • Se pueden intercambiar los símbolos.

• Para anidar una lista hay que añadir cuatro espacios en blanco antes del símbolo de lista.

▪ Ordenadas. Se usa número + punto + espacio antes del elemento de la lista. Se puede usar siempre el mismo número.

• Se pueden anidar listas igual que desordenadas, cuatro espacios.

✗ Bloque. Para crear un bloque o remarcar una zona hay que usar tres virgulillas (alt + 126), ~~~, antes y después del texto del texto del bloque.

✗ Regla horizontal. Tres asteriscos, guiones o guiones bajos. (\*\*\*, ---, \_\_\_₎. Pueden estar juntos o separados. (\* \* \*, - - - , \_ \_ \_). • Elementos de línea:

✗ Énfasis. Se puede, antes y después de las palabras a remarcar, usar asteriscos, \*, o guiones bajos, \_.

✗ Cursiva. Un único asterisco, \*, o guion bajo, \_.

✗ Negrita. Dos asteriscos, \*\*, o guiones bajos, \_\_.

✗ Ambos. Tres asteriscos, \*\*\*, o guiones bajos, \_\_\_.

✗ Enlaces. La palabra del enlace se pone entre corchetes, [], mientras que el enlace se pone entre paréntesis, ().

▪ Se puede hacer un enlace de referencia, similar a hover, usando palabras de referencia, que puede estar en otro sitio del texto, con formato [referencia]: <http://www.sitiodelenlace.com>.

▪ Si se quiere ver la ruta del enlace, la url, en vez de la referencia, hay que usar ángulos, <>, en la ruta del enlace.

✗ Código, similar a la etiqueta de html. Se encierra el texto entre comillas sencillas, `` (la de al lado de la p).

✗ Texto preformateado, similar a la etiqueta

. Empezar cada línea del texto con cuatro espacios.

✗ Imágenes. Usar exclamación, !, antes de un enlace a una imagen.

![Texto alternativo](/ruta/a/la/imagen.extension).

▪ Texto alternativo. Usar comillas, “, tras la ruta de la imagen.

Si necesitamos usar un carácter que Markdown usa para formatear el texto como asteriscos o guiones, debemos usar la barra invertida, \.

• @author. Solo clases e interfaces. Nombre del autor de la clase que se documenta.

• @deprecated. La clase ha sido reprobada y no debería seguir usándose. Puede haber varias razones como problemas de seguridad o uso de la clase, haberla escrito con malas prácticas o que haya una nueva versión más actualizada. Normalmente se usa @see para indicar la nueva opción válida.

• @exception. Igual que @throws aunque definió antes y se considera que es mejor o más exacto usar @throws.

• @param. Solo métodos y constructores. Define un parámetro de entrada.

• @return. Solo métodos. Describe el valor de salida del método.

• @see. Enlace o referencia a otro sitio, ya sea clase, método, guía,…

• @serial. Describe el motivo del campo y sus posibles valores. También puede usarse @serialField o @serialData.

• @since. Indica a partir de qué versión se ha añadido el componente.

• @throws. Excepción que puede lanzar el método y hay que tener en cuenta.

• @version. Solo clases e interfaces. Número de versión de la clase. Es decisión de los programadores establecer un sistema de versiones, pero una vez establecido no deberían modificarlo.

El **orden recomendado** para poner las etiquetas es

@author (solo clases y requerido),

@version (solo clases y requerido),

@param (solo métodos),

@return (solo métodos),

@exception (o su sinónimo @throws),

@see, @since, @serial, @deprecated.