**Git y GitHub**

Comandos en Git (Consola de git).

* **git status:**

Verifica el estado actual de los archivos que estoy trabajando con git. Muestra los archivos que han sido modificados o eliminados, de manera que informe al usuario lo que ha pasado.

* **git add <archive>**

Selecciona un archivo para realizar un commit

* **git commit**

Abre una ventana para escribir el título del commit

* **git commit –m “message”**

Agrega el título al commit directamente

* **git reset --soft HEAD­­­~1**

Deshace el último commit realizado.

* **git push**

Agrega un repositorio a la nube o aun repositorio remoto: Ej. GitHub

* **git remote**

Obtiene el nombre de la dirección del repositorio remoto

* **git remote –v**

Obtiene la Url del repositorio remoto

* **git push origin master**

Sube el repositorio a GitHub los parámetros son remoto=origin y el rama=master

* **git pull**

Descarga los archivos del remoto

* **git pull origin master**

Obtengo los datos que hay en el repositorio en la rama master remoto al pc

* **git commit .am “message”**

Seleccionó todos los archivos que he modificado localmente para realizarle un commit

* **git branch**

Obtiene todas las ramas creadas en el proyecto

* **git branch <name>**

Creo una nueva rama con el nombre que quiera

* **git checkout <name>**

Cambio de una rama a otra pasándole el nombre de la rama que quiero estar

* **git checkout –b <name>**

Creo nueva rama y me coloca en esa nueva rama

* **git log –-graph –-decorate --oneline –all**

Muestra de forma clara los commits realizados en cada rama, sin información extra, solo con el texto.

* **git log**

Muestra el historial completo de commits realizados

* **git log --oneline**

Muestra el historial con el código del commit y el nombre

* **git log --oneline –decorate**

Muestra el historial con el código corto y el nombre clasificados por la rama

* **git merge <name-rama>**

Trae los datos de la rama con el nombre que le he dado a la rama actual en la que esté, creando una nueva mezcla

* **git branch –d <name-branch>**

Elimina una rama que no vaya a utilizar

**-d** comprueba que no vayas a perder el trabajo

**-D** git no comprueba las ramas a eliminar, no comprueba la mezcla que haya entre una rama secundaria y la rama principal

* **git gc**

Elimina todos los archivos basura, para liberar espacio en el disco.

* **git checkout <code-commit>**

Lleva a un commit atrás en el tiempo (detached HEAD), muestra los archivos que estaban en ese commit

* **git checkout master**

Vuelve a la rama principal apuntando al último commit.

**Tags**

Sirven para colocar un alias al commit, y así poder buscarlo por ese nombre

* **git tag <version-name>**

Crea una etiqueta para el último commit de la rama

* **git tag <version-name> <commit-code>**

Crea una etiqueta para ese commit en específico

* **git tag**

Muestra el listado de Tags en el repositorio

* **git checkout <tag>**

Vuelve a un estado anterior del código donde se haya colocado un tag

**Nota:** Se pueden agregar como repositorio remoto a un repositorio que esté en el ordenador, y hacerle push a ese repositorio

* + git remote add origin -/Dropbox/Repositorio.git
  + git push origin master
* **git branch --all**

Muestra todas las ramas incluyendo las remotas

push -> Envía

pull->Descarga

commit -> Anotación

issue -> Publicación, errores o problemas.

branch -> Rama Es una extensión del proyecto principal con sus propios commit y archivos, los puede utilizar para la creación de nuevas características del proyecto sin afectar el original

merge -> Mezclar, fusionar o Unir, ramas en git.

pull -> Jalar o traer un repositorio que esta remoto.

push -> Empujar, presionar o subir un repositorio trabajado localmente a un servidor remoto.

Al realizar commit puedo hacer referencia a issues colocando la almodilla o numeral

#1 junto con el número que hacer referencia al issue tratado

Para cerrar un issue, se puede hacer desde GitHub o desde la terminal cuando se realiza un commit colocando la palabra **Closes #1** en la descripción del commit

Con Fork, puedo traer un repositorio de otra persona para poder utilizarlo como propia y realizar cambios.

* git clone -> clona un reopsitorio en el computador
* $ git add –p <archive>

Preguntá que cambiós quiere que se vea en el historial der archivo, luego cuando se haga el commit

Los issues y los pull request son importante para la colaboración de proyectos en GitHub

Wiki en GitHub -> Sirve para crear documentación sobre algún proyecto.

Con wiki escribe, artículos, documentación acerca del proyecto.

* Crear servidor con python

$ python –m simpleHTTPServer 8000

Crea un sevidor en el puerto 8000

**Cambiar el nombre del repositorio**

En la sección settings del repositorio puedo realizar cambios acerca del repositorio que se está trabajando.

**GitHub Pages**

Permité guardar páginas estáticas, crea una nueva rama en GitHub, pero al cambiar el código o la información de la rama master, se actualiza automáticamente en la página que hemos creado.

Se puede crear una página personal también.

**Páginas personales**

Para crear una página personal con dominio de GitHub, se debe crear un nuevo repositorio con el nombre de usuario que tenemos en GitHub y agregar github.io

* jocode.github.io

De esta manera el sistema lo reconoce como página web personal

En caso de usar un dominio de terceros

Crear un archivo cname así como suena, y redirigir al dominio

Y en el panel de control del dominio crear una archivo cname con las siguientes características

Crear archivo cname, y dentro de él colocar las siguientes líneas para apuntar a donde está la página

nombredeusuario.github.io.

[www.nombredeusuario.github.io](http://www.nombredeusuario.github.io).

**Releases**

Son las diferentes versiones que se realiza a un software, GitHub realiza estas contribuciones

**Gits**

Permite crear snipets de código para compartir con otras personas

**Cosas que se debe investigar**

* Claves SSH
* Git LFS Large File Storage