Tabla

Descripción generada automáticamente con confianza baja

Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

INSTALANDO Git

**Git**

Si tienen Windows, pueden seguir el siguiente link que los llevará al instalador de Windows: [Instalador de Git para Windows](https://git-scm.com/download/win).

Si tienen Mac, aquí tienen las instrucciones de instalación: [Instrucciones para instalar Git en Mac](https://git-scm.com/download/mac).

Y si tienen Linux, el siguiente link les ayudará: [Instrucciones para instalar Git en Linux](https://git-scm.com/download/linux).

Captura de pantalla de un celular

Descripción generada automáticamente

La ventana de comandos puede recibir comandos tanto de Windows como de Linux por ejemplo para visualizar un directorio y sus carpetas utilizamos la instrucción dir.

Texto

Descripción generada automáticamente

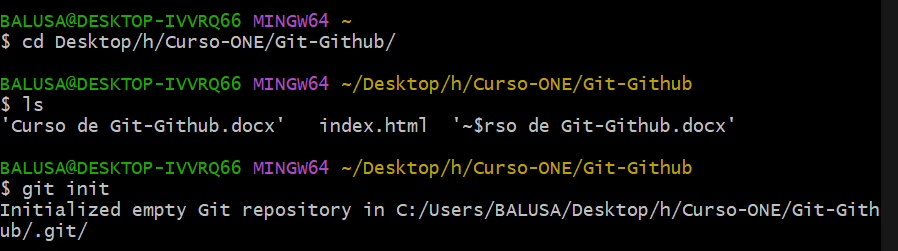
Con Linux utilizamos la instrucción ls

Texto

Descripción generada automáticamente

Repositorios

Utilizamos el comando **“cd”** junto a la dirección exacta de nuestra carpeta, así como se muestra en la imagen, para poder visualizar los archivos dentro de la carpeta utilizamos **“ls”.** Por último, la instrucción **“git init**” inicializa nuestro repositorio en la carpeta antes mencionada



Para asegurarnos que el repositorio se generó con éxito utilizamos la instrucción “**git status”.**

El texto desplegado nos informa que no hemos agregado documentos para que estos sean monitoreados y para agregar los mismos debemos colocar **“git add” <nombre\_del \_archivo>**

Texto

Descripción generada automáticamente

Captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

**LECCIÓN 2 INICIANDO LOS TRABAJOS**

2.1 Guardando las modificaciones

Para agregar los archivos de nuestra carpeta a nuestro repositorio se utiliza la instrucción **“git add”** acompañada del nombre del archivo o como en el caso de abajo si colocamos un punto le decimos a la máquina que agregue todos los archivos de la carpeta.

Una vez más verificamos el estatus del git y podemos ver que los archivos aparecen de color verde esto indicándonos que se han agregado con éxito. Texto

Descripción generada automáticamente

Para generar un punto de modificación en el historial del archivo utilizamos **(commit -m** “descripción del cambio”)**.** Si no tienes configurado el git obtendrás el texto que aparece abajo**.** Para poder configurarlo debemos utilizar las 2 instrucciones que nos muestra **(git config – global user.email “Escribe tu email”)**

**(git config – global user.name “Escribe tu nombre”)**

Texto

Descripción generada automáticamenteUna vez hecho esto estará configurado el git

Ya configurado el git ahora si podemos crear en punto de guardado, y checar una vez más el status del git mostrándonos que no ha ocurrido ninguna modificación a los archivos.

Probamos que el repositorio esta siendo vigilado modificamos el código y volvemos a verificar el status, si todo esta correcto aparecerá la leyenda que el archivo index ha sido modificado.

Texto

Descripción generada automáticamente

Con la modificación hecha agregamos el archivo index y checamos el status para asegurarnos que este se agregó correctamente.

Por último, creamos un punto de modificación agregando una descripción del cambio realizado con el **“git commit -m”** y si todo salió bien aparecerá un mensaje diciendo que 1 archivo fue cambiado, 1 línea fue insertada y 1 línea fue eliminada.

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente con confianza media

**2.2 Viendo el histórico**

Para ver el historial de nuestras modificaciones o commit ejecutamos la instrucción “git log” el cual nos mostrara:

1.- Commit y su hash 🡪 Un hash es una identificación única, no pueden existir dos commit con el mismo hash.

2.- HEAD es el lugar donde estamos posicionados en ese momento.

3.- master es el branch o rama.

4.- autor y fecha de realización o modificación del archivo.

Texto

Descripción generada automáticamente

Existen 2 maneras de configurar nuestro git y son:

“git config --local” 🡪 Crea la configuración de manera local ósea que esta solo funciona para este repositorio.

“git config –global” 🡪Crea la configuración de manera global ósea que la misma servirá para cualquier repositorio de aquí en adelante en la misma computadora.

Para configurar el nombre que se utiliza utilizamos **(git config –local user.name “”).** Para visualizar el nombre o correo utilizamos “git config user.name o user.email”.

Texto

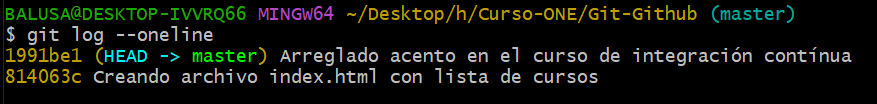
Descripción generada automáticamente

Existen distintas formas para visualizar los datos históricos de nuestros commit una de ellas es ver los datos en una línea con la instrucción **“git log –oneline”.**

En cada línea tenemos la siguiente información:

1.- Un resumen del hash

2.- El Branch y la información que nosotros colocamos.



Si queremos más información podemos ejecutar **“git log -p”**, la cual nos muestra con más detalle la línea modificada.

Texto

Descripción generada automáticamente

Cuando tenemos las líneas de color verde como en la imagen quiere decir que todas estas líneas de código fueron añadidas al repositorio.

Para poder salir de este historial podemos poner **“: q”** para regresar a la consola de git.

Texto

Descripción generada automáticamente

Existe una página que nos muestra distintas maneras de visualizar el histórico de nuestros commit y esa es <https://devhints.io/git-log-format>.

Una forma diferente es **(git log –pretty=”format:%h %s”)** al ejecutarse el %h nos muestra un resumen del hash y el %s nos muestra el mensaje. Tambien existe un comando llamado **“git log –help”** el cual nos mostrara una página con ayuda.

Texto

Descripción generada automáticamente

2.3 Ignorando archivos

Si queremos que git no monitoree todos los archivos existe la posibilidad de crear un archivo para ignorar estas carpetas o archivos y se hace de la siguiente manera, creamos un archivo en la carpeta llamado **“.gitignore”** dentro del cual colocamos el nombre de la carpeta y nombre del archivo que deseamos ignorar.

Interfaz de usuario gráfica

Descripción generada automáticamente

Una vez hecho esto debemos agregar el archivo a nuestro repositorio junto con su commit y el mensaje.

Después verificamos el status del repositorio, si todo esta bien y no falta nada para ser arreglado podemos pasar a checar el histórico de los commit y observar que todo fue guardado correctamente.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

LECCIÓN 3 COMPARTIENDO EL TRABAJO

3.1 Repositorios remotos

Para poder crear un repositorio remoto primero debemos crear un servidor donde alojaremos todos los archivos para que se pueda acceder fácilmente a ellos.

1.- Primero utilizamos **“cd ..”** para que subamos un nivel dentro de nuestras carpetas.

2.- Para crear una carpeta utilizamos **“mkdir servidor”** o nombre de la carpeta.

3.- Ahora entramos a la carpeta con el comando **“cd servidor/”.**

4.- Dentro de la misma ejecutamos un **“git init --bare”** la cual nos indica que este repositorio es puro lo que quiere decir que solo contienen las modificaciones de los archivos.

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

Ahora regresaremos a nuestra carpeta de repositorio principal, una vez dentro de la misma tenemos que decirle que existe un repositorio remoto así que ejecutamos un **“git remote”.** Una vez hecho esto enumerara los servidores remotos, pero aún no conoce ninguno no aparecerán. Para agregar nuestro servidor ejecutamos **“git remote add servidorlocal dirección de la carpeta servidor”.**

Una vez agregado este servidor local volvemos a verificar con el comando git remote y ahora si aparecerá enlistado nuestro servidor.

Si queremos asegurarnos de que nuestro servidor esta conectado a la ruta correcta ejecutamos el comando **“git remote -v”,** el cual nos mostrara dos resultados un **(fetch)** y un **(push).**

**Fetch** 🡪 Es el lugar donde nosotros obtendremos la información del servidor remoto.

**Push** 🡪 Es el lugar al que nosotros vamos a enviar información.

Para este caso en particular el camino será el mismo tanto para fetch y push pero pueden existir distintas rutas.

Texto

Descripción generada automáticamente

Como ejercicio para entender el servidor remoto y como otras personas pueden interactuar con nuestro repositorio hicimos lo siguiente:

1.- Subimos un nivel en nuestra ruta y una vez allí creamos una carpeta llamada Ana con la instrucción **“mkdir Ana”.**

2.- Entramos a la carpeta de Ana con **“cd Ana/”.**

3.- Dentro de la misma ejecutamos un **“git clone ruta completa del servidor”,** si dejamos la instrucción así la nueva carpeta tendrá el mismo nombre que la carpeta original ósea servidor, para que esto no sea así damos un espacio y colocamos nuevo nombre de la carpeta en este caso se llamara proyecto.

Texto

Descripción generada automáticamente con confianza baja

Texto

Descripción generada automáticamente

El archivo creado estará vacío ya que nuestro servidor remoto no tiene nada así solo se creo la conexión con nuestro servidor.

Abajo se muestran todos los archivos creados desde la terminal de **git bash.**

Captura de pantalla de un celular

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

3.2 Sincronizando los datos

Los pasos que seguir para poder enviar y recibir los datos de un servidor loca son los siguientes. Una vez dentro de nuestra carpete de repositorio principal.

1.- Ejecutamos “git push servidorlocal master” 🡪 Esta instrucción empuja o hace un push de nuestro servidor local acompañado de nuestro branch master que por el momento es el único que tenemos.

Texto

Descripción generada automáticamente

2.- Regresamos a la carpeta de Ana ósea de la persona que quiere obtener los archivos de nuestro servidor, dentro de ella podemos notar que aún no existe ningún archivo en su carpeta.

Escala de tiempo

Descripción generada automáticamente con confianza baja

3.- Para ver los repositorios que tiene Ana utilizamos un git remote el cual nos dice que el repositorio se llama origin que es el nombre por defecto de git.

4.- Ahora renombraremos el repositorio con un **“git remote rename origin servidorlocal**”.

Texto

Descripción generada automáticamente

5.- Corroboramos el nombre con git remote.

6..- Ahora debemos jalar (pull) los elementos del servidor al repositorio de Ana con la instrucción “git pull servidorlocal master”, podemos notar que al realizar esto se creo una nueva Branch llamada master dentro del proyecto de Ana.

7.- Para corroborar si los elementos se obtuvieron observamos los elementos de la carpeta.

Texto

Descripción generada automáticamente

Podemos observar que los elementos se copiaron de la mejor manera

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Probamos que todo esta bien y hacemos una modificación en el index de Ana para demostrar que si ella hace cambios puede subirlos al servidor y nosotros copiarlos

Una vez realizados los cambios verificamos que git los noto con git status, esto nos mostrara que el archivo index.html fue modificado.

Ahora debemos agregar este archivo con “git add index.html” y su respectivo commit (git commit -m “Renombrando curso de Integración continúa”). Si todo esta correcto nos dira que un archivo fue cambiado y hubo líneas de inserción y eleminadas.

Texto

Descripción generada automáticamente

Como siguiente paso ahora Ana debe empujar(push) el archivo al servidor para que este pueda ser descargado por terceros.

Texto

Descripción generada automáticamente

Para obtener el archivo modificado ahora debemos ir a nuestro repositorio y jalar (push) el elemento con “git pull servidorlocal master”

Texto

Descripción generada automáticamente

Por ultimo verificamos el histórico con información detallada con “git log -p” y confirmamos que la modificación y copiado del repositorio fue realizado con éxito.

Texto

Descripción generada automáticamente