Ha llegado el momento de que pongas en práctica lo visto en clase. Para hacer esto, sigue los pasos que se enumeran a continuación.

1) En la terminal (o **Git Bash**, en Windows) navega hasta la carpeta recién creada (usa el comando cd para navegar entre carpetas);

2) Ejecuta el comando git add index.html para marcar el archivo para ser guardado (commiteado);

3) Ejecuta git status y verifica que el archivo haya cambiado de estado y esté listo para ser guardado (commiteado);

4) Después de agregarlo, ejecuta el comando git commit -m "Creando archivo index.html con lista de cursos". Puedes cambiar el mensaje de commit si quieres;

5) Cambia el archivo **index.html**. Agrega el acento en "Integración continua", por ejemplo;

6) Agrega el archivo a guardar con git add .;

7) Ejecuta el comando git commit -m "Acento agregado en el curso de Integración Continua". Puedes cambiar el mensaje de commit si quieres;

8) Ejecuta el comando git log y analiza su salida. También ejecuta git log --oneline, git log -p y otras alternativas que desees probar;

9) Crea un archivo vacío con el nombre que desees, por ejemplo, ide-config;

10) Crea el archivo **.gitignore** y agrega una línea con el nombre del archivo recién creado (ide-config, en el ejemplo anterior);

11) Ejecuta git status y verifica que el archivo **ide-config** no esté en la lista para ser agregado;

12) Agrega (con git add .gitignore) y confirma (con git commit -m "Adding .gitignore") el archivo **.gitignore**.

* Que un commit es la forma de guardar un estado o versión de nuestro código;
* Cómo agregar archivos para *commitear* con git add;
* Cómo *commitear* archivos usando el comando git commit;
* Cómo verificar el historial de *commits*, a través de git log y algunas de sus opciones:
  + git log --oneline
  + git log -p
  + git log --pretty="parámetros de formato"
* Cómo hacer que Git no monitoree archivos, a través de **.gitignore**
* Que no debemos realizar *commits*, es decir, guardar un estado, de nuestra aplicación que no esté funcionando.

1) Cree una nueva carpeta en su computadora;

2) En la terminal (o **Git Bash**, en Windows) navega hasta la carpeta recién creada (use el comando cd para navegar entre carpetas);

3) Ejecuta el comando git init --bare;

4) Navega hasta la carpeta donde se encuentra tu proyecto;

5) Ejecuta el comando git remote add servidorlocal {ruta}. Reemplaza {ruta} con la ruta completa de la carpeta recién creada;

6) Crea una nueva carpeta en tu computadora, para representar el trabajo de otra persona;

7) En la terminal (o **Git Bash**, en Windows) navega hasta la carpeta recién creada;

8) Ejecuta el comando git clone {ruta} proyecto. Reemplaza {ruta} con la ruta completa a la carpeta que creamos en el primer paso;

9) Ten en cuenta que el repositorio clonado está vacío;

10) Accede a la carpeta del Proyecto y ejecuta el comando git remote rename origin servidorlocal para cambiar el nombre del repositorio local de la otra persona de "origin" a "servidorlocal";

11) Navega a la carpeta donde se encuentra tu proyecto original;

12) Ejecute el comando git push servidorlocal master para enviar tus cambios a tu servidor;

13) Navega hasta la carpeta creada en el paso 6;

14) Ejecuta el comando git pull servidorlocal master para descargar los cambios;

15) Abre tu navegador y vé a <http://github.com/>;

16) Crea una cuenta;

17) Crea un nuevo repositorio haciendo clic en el símbolo más (+) en la esquina superior derecha;

18) En la terminal (o **Git Bash**, en Windows) agrega el repositorio remoto recién creado a tu proyecto inicial (los comandos los muestra el propio GitHub);

19) Ejecuta git push origin master para enviar tus cambios al repositorio en GitHub.

* En esta aula aprendimos: Lo que son los repositorios remotos
* Cómo crear un repositorio de Git sin una copia de los archivos (con --bare) para usar como servidor;
* Cómo agregar links a repositorios remotos, con el comando git remote add;
* Cómo descargar un repositorio por primera vez clonándolo con el comando git clone;
* Cómo enviar nuestros cambios a un repositorio remoto, con git push;
* Cómo actualizar nuestro repositorio con los datos en el repositorio remoto, usando git pull;
* Qué es **GitHub** y para qué sirve;
* Cómo crear un repositorio en **GitHub**;
* Cómo agregar un repositorio de **GitHub** como repositorio remoto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Git remote remove origin - remueve el origin

O error 403 niega el acceso

En windows panel de control / windows manager/ generic credential / remove github credentials

