

**DESARROLLO DE APLICACIONES WEB AVANZADO**

**Version Control:**

**Comandos básicos de GIT**

**Version Control**

**Git**

**Objetivos:**

Al finalizar el laboratorio el estudiante será capaz de:

* Entender el funcionamiento de Git
* Utilizar control de versiones en todos los proyectos posibles
* Implementación correcta de Git de forma colaborativa

**Seguridad:**

* Ubicar maletines y/o mochilas en el gabinete del aula de Laboratorio.
* No ingresar con líquidos, ni comida al aula de Laboratorio.
* Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

**Equipos y Materiales:**

* Una computadora con:
* Windows 7 o superior
* VMware Workstation 10+ o VMware Player 7+
* Conexión a la red del laboratorio
* Máquinas virtuales:
* Windows 7 Pro 64bits Español - Plantilla
* Instalador de node.js

**Procedimiento:**

**Lab Setup**

1. **Instalación de Git**
   1. En caso cuenta con un Sistema Operativo con Shell de Linux, ejecute el siguiente comando



* 1. En caso no cuente con dicha terminal, utilice el siguiente enlace para descargar el instalador de Windows (en caso tenga Windows con la terminal de Linux, utilice preferiblemente el paso anterior)

<https://gitforwindows.org/>

* 1. Después de haber realizado la instalación, ya tendremos git incorporado a nuestra consola de comandos.

¿Qué es Version Control?

Para casi todo proyecto, su código fuente es el recurso o activo más preciado que puede poseer. El código es una expresión invaluable de conocimiento, la forma de plasmar la lógica de negocio y el cómo los desarrolladores han logrado hallar soluciones para optimizar el flujo de la misma. Version Control protege a los desarrolladores de la catástrofe y de la degradación casual del error humano y sus ilimitadas consecuencias.

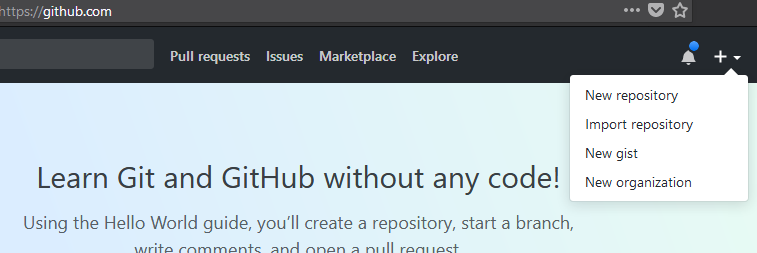
El problema generalmente viene cuando el proyecto crece y ya no es uno, sino un equipo de desarrolladores. Desarrollador A y Desarrollador B están trabajando en características distintas y distintos archivos y ninguno debería afectar al otro en su trabajo.

Version Control ayuda a equipos de software a resolver estos problemas, llevando un control individual de cada uno de estos cambios por cada usuario y ayudando a prevenir posibles conflictos entre cada cambio.

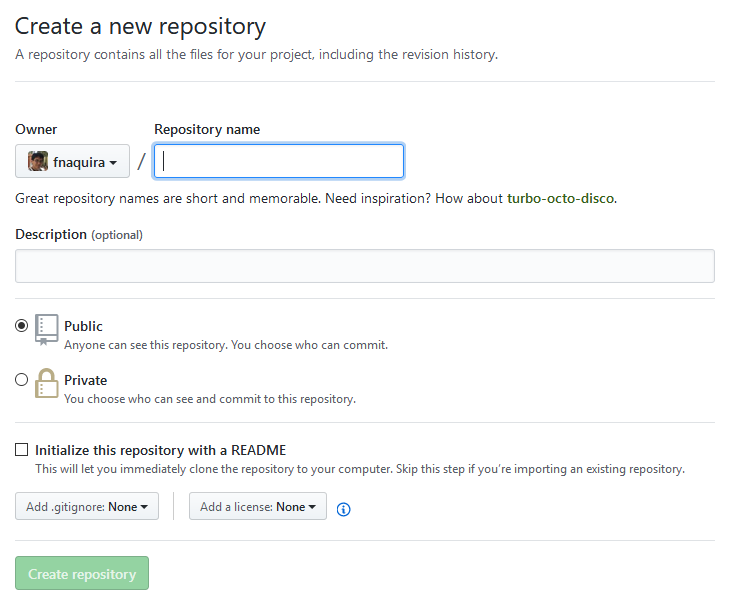
Git almacena la información como instantáneas del proyecto a lo largo del tiempo.



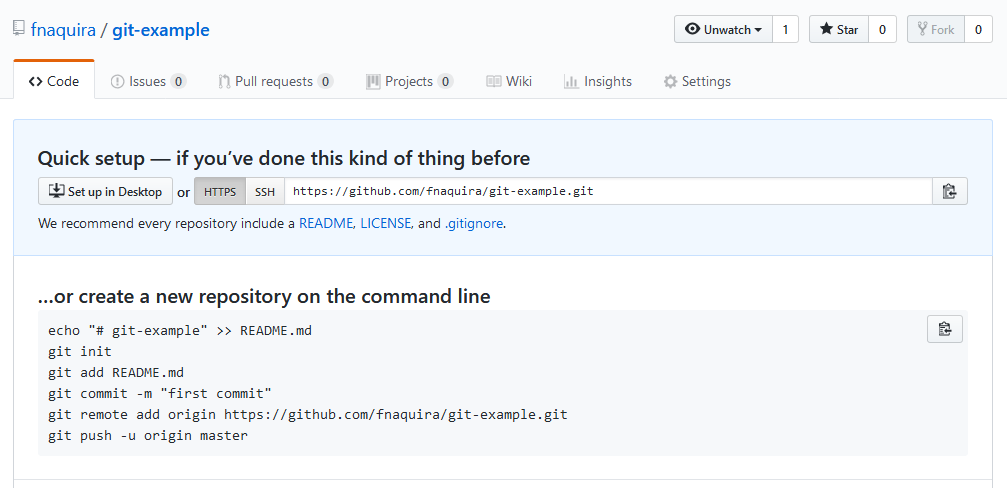
1. **Creación de un repositorio.**
   1. Este procedimiento puede variar de acuerdo a la empresa que una escoja para alojar el código. Hay varias alternativas como Github, Bitbucket, Gitlab, etc. Sin embargo todas tienen un procedimiento similar y al final los comandos de git son los mismos. En nuestro ejemplo crearemos un repositorio en Github.
   2. En la página de Github, luego de haber iniciado sesión



* 1. Veremos el siguiente formulario que nos pedirá la configuración inicial de nuestro repositorio. Ingresemos el nombre **git-example** y hagamos click en **Create repository**



* 1. Posteriormente, veremos

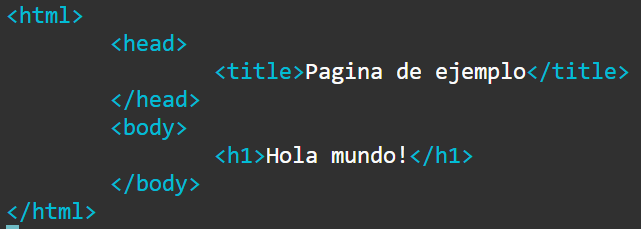


* 1. Su proyecto ha sido creado con éxito. Ahora tenemos dos posibles caminos de acción:
     1. Sincronizar nuestro código local con el repositorio remoto
     2. Clonar nuestro nuevo proyecto

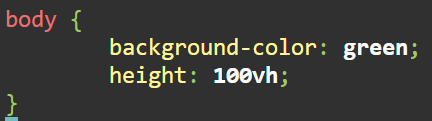
1. **Sincronizar nuestro código local con el repositorio remoto**
   1. Nos situaremos con la consola en la carpeta de nuestro proyecto. En nuestro ejemplo, yo me situaré en la carpeta pagina\_ejemplo, que es el nombre de mi página web.

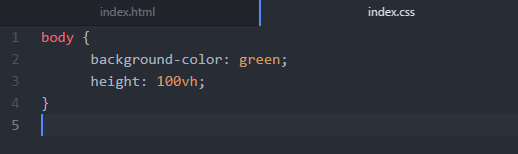


* 1. Algo que se puede notar, es que la carpeta elegida no comparte el nombre del repositorio, por lo que vemos que no es obligatorio que ambos coincidan.
  2. En mi carpeta tengo los siguientes archivos de prueba:
     1. index.html



* + 1. index.css





* 1. Ejecutaremos el comando para iniciar un proyecto git



* 1. Obtendremos una respuesta como la siguiente

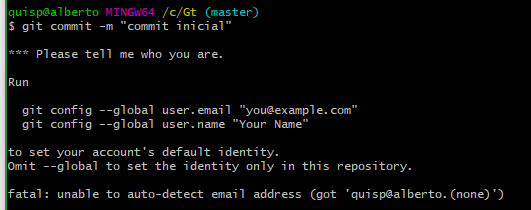


* 1. Procedemos a declarar que tenemos archivos para agregar a nuestro repositorio. Los agregamos con el comando git add -A

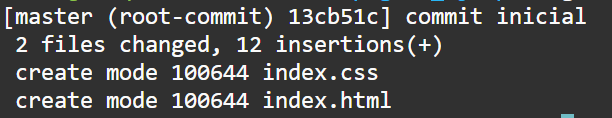


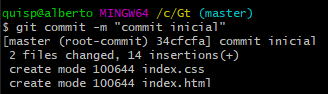
* 1. Después de haber agregado nuestros archivos, procedemos a crear





* 1. Después de haber eralizado el commit inicial, veremos un mensaje de nuestra acción





[**git-dash**](https://github.com/alberthuarcaya/git-dash)

* 1. Como es un repositorio nuevo, debemos decirle cuál será su conexión remota. Esto lo logramos con el siguiente comando (solamente variará la url de acuerdo a su repositorio)



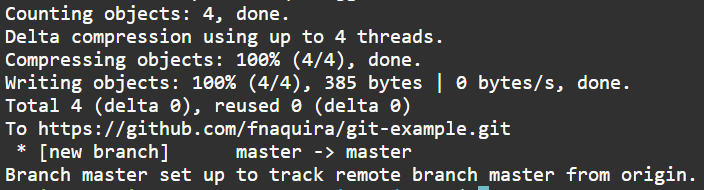
* 1. Depués de haber inicializado el repositorio y haberle indicado su dirección remota, procedemos a hacerl un push, o un empuje de los archivos locales al origen remoto (por defecto, llamado **master**)



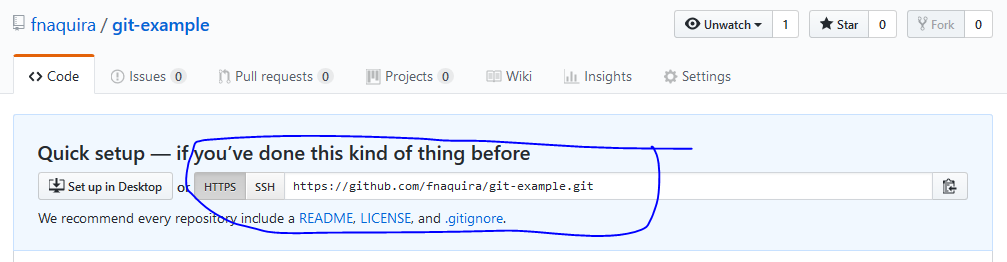
* 1. Será común que Git nos solicite nuestras credenciales para asegurar que tenemos permisos de escritura sobre el proyecto que indicamos. Basta con colocarlas según sea requerimiento



* 1. Finalmente, veremos un mensaje de éxito en que nos describe la subida de los archivos



1. **Clonar nuestro nuevo proyecto**
   1. Otra opción más simple es clonar el nuevo repositorio y copiar el código adentro.
   2. En la interfaz de Github, veremos en la parte superior la siguiente imagen, de la cual tomaremos la URL del repositorio



* 1. En nuestra consola, dentro de una carpeta fuera de un repositorio git (debemos cuidarnos de este detalle), usaremos el comando de clonar repositorio.



* 1. Ahora copiaremos nuestros archivos dentro de la carpeta generada. Otra forma de clonar un repositorio git es indicándole el nombre de nuestra carpeta destino. Para esto, basta con indicarle el nombre deseado al final del comando.



* 1. Procedemos a decirle a git que agregue nuestros archivos



* 1. Crearemos el commit inicial



* 1. Finalmente, ejecutaremos git push, para subir todos los archivos a la nube



* 1. Esta forma es más sencilla de realizar, por lo que les recomiendo siempre ejecutarla.

1. **Subir cambios a nuestro repositorio**

Esto es lo más importante a recordar acerca de Git si quieres que el resto de tu proceso de aprendizaje prosiga sin problemas. Git tiene tres estados principales en los que se pueden encontrar tus archivos: confirmado (committed), modificado (modified), y preparado (staged). Confirmado significa que los datos están almacenados de manera segura en tu base de datos local. Modificado significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos. Preparado significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

Esto nos lleva a las tres secciones principales de un proyecto de Git: el directorio de Git (Git directory), el directorio de trabajo (working directory), y el área de preparación (staging area).

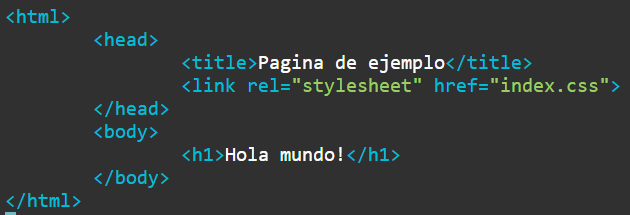


El equivalente de stage files es git add –A

El equivalente de commit es justamente git commit

Decimos que un proyecto ha recibido un checkout o está libre de cambios después de hacer un git push

* 1. Nuestro archivo html está en estado básico y ni siquiera está agregando la hoja de estilos index.css, por lo que vamos a modificarlo.



* 1. Procedemos a hacer un stage de los archivos, declararlos para el proyecto



* 1. Depués de haber hecho el stage, debemos hacer otro commit o “instantánea” del proyecto. A diferencia de nuestro commit inicial, variará el mensaje entre comillas y esto se debe a que este mensaje es personalizable. Podemos repetir siempre el mismo mensaje pero no será útil para poder rastrear el proyecto hacia un estado determinado, por lo que es importante colocar mensajes claros del cambio que se desea comprometer, o mejor dicho, hacer commit



* 1. Este commit tiene una respuesta algo distinta. Nos indica que ha detectado un archivo modificado (nuestro index.html)



* 1. Ahora solamente hagamos push y nuestro cambio se reflejará en la nube.

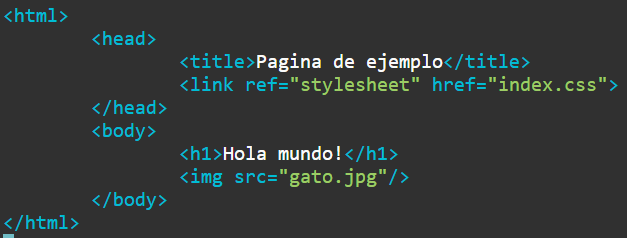


1. **Actualizar nuestro código local con los cambios del repositorio (trabajo en comunidad)**
   1. El fin de utilizar git, aparte de poder llevar un verdadero control de versionado de nuestro código, es el de facilitar el trabajo colaborativo. Para simular el estar trabajando con otra persona, nos moveremos a un nivel atrás de nuestro git, procederemos a clonarlo y generar cambios en el código (tal como haría algún compañero nuestro)





* 1. Pegaremos la imagen gato.jpg en nuestro nueva carpeta y modificaremos el archivo index.html (el docente le proveerá dicha imagen, pero usted puede utilizar esa o alguna descargada de internet)



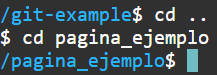
* 1. Procedemos a subir este cambio, tal como hemos aprendido



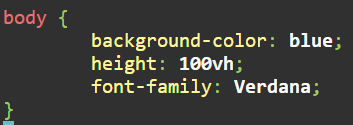




* 1. Ahora volveremos a nuestra carpeta original y haremos otro cambio distinto.



* 1. Modificaremos el archivo index.css para tener un fondo distinto y también otra fuente

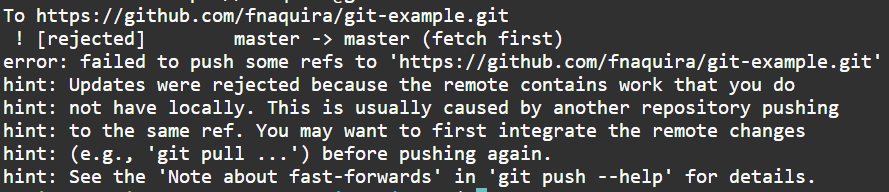


* 1. Procedemos a hacer el commit correspondiente y a subir nuestros cambios a la nube





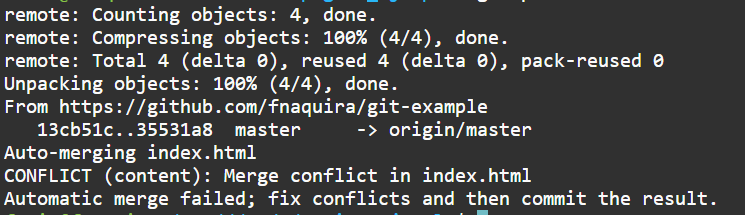
* 1. Debido a que en nuestra simulación, otro colaborador del proyecto ha subido código adicional, e inclusive, nuevos archivos, nos aparecerá el siguiente error



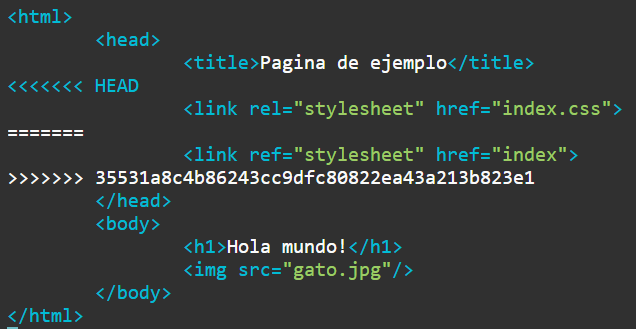
* 1. Esto indica que hay cambios en el repositorio que no tenemos en nuestra rama local. Para poner al día nuestra rama local, bastará con jalar o “pullear” los cambios del repositorio



* 1. Muchas veces, esto será más que suficiente para luego poder insistir con git push. En ocasiones tendremos un mensaje como el siguiente



* 1. Cada vez que veamos un CONFLICT en mayúsculas, significa que debemos revisar ese archivo ya que Git no ha sido capaz de conciliarlo. Veamos el archivo index.html para ver el conflicto.



* 1. Git resalta mediante los señaladores <<<<<<< HEAD y ===== lo que había en el repositorio remoto, mientras que lo siguiente es lo que hay en nuestro repositorio local. Podemos ver que no tuvo problemas en mezclar el contenido de <img src=”gato.jpg” /> debido a que este no interfería con ninguna parte clave y podía ser fácilmente mezclado. Lo único que deberemos hacer es borrar las líneas inútiles y decidir con que versión nos quedaremos, si la remota o la local.
  2. Después de haber resuelto este conflicto, bastará con utilizar nuevamente los comandos git add, git commit y git push para poner al día nuestro repositorio remoto.

1. **Finalizar la sesión**
   1. Apagar el equipo virtual
   2. Apagar el equipo