Git cuenta con 3 estados de trabajo

* El “Working Directory”: es donde estarás trabando con todos tus archivos.
* “Staging área”: es donde agregaras todos los archivos para el guardado, aquí es donde se ve el manejo de versiones de git.
* “Repository”: una vez realizados todos los cambios se pasa todo a un repositorio que genera la versión de trabajo que estemos realizando.

Comandos básicos. (todos se deben escribir en minúsculas)

* Git init: es para iniciar un proyecto, por más que ya tengamos un proyecto realizado o en proceso de trabajo, lo que hará es iniciar el uso de git en el mismo.
* Git add (nombre del archivo): envía los archivos del Working Directory al Staging área.
* Git status: es para ver el estado del archivo, si no se pasó al Staging área o si ya está listo.
* Git commit: pasa los archivos de Staging área, al Repository, crea la primera imagen de la versión del código.
* Git push: se usa para subir el archivo a un repositorio remoto.
* Git pull: te traerá los cambios que se hayan realizado sobre ese archivo, que estén guardados en el repositorio remoto.
* Git clone: clona el repositorio remoto al pc

Configurando git.

Para aprender a usarlo correctamente se debe más que nada practicar, para comenzar se descargara desde [www.git-scm.com](http://www.git-scm.com), realizamos la instalación normal, al tener la ventana de “Select Components” verificamos si la opción “Git bash here” y “Git GUI here” deben estar tildados, en la ventana de “Adjusting your PATH enviromment” tildaremos la opción “Use Git and opcional unix tool from the Windows command prompt”. Con ello quedaría configurado correctamente para poder usar Git

Comenzando a usar Git.

Una vez configurado e instalado, podemos comenzar a usarlo en nuestros proyectos, para ello nos dirigimos a la carpeta donde tengamos un proyecto realizado o estemos por realizar uno, hacemos click derecho y seleccionamos la opción “Git Bash Here”. En la pantalla de comandos que se abrirá debemos verificar que la dirección que muestre donde se está ejecutando sea la de nuestra carpeta donde está el proyecto, luego de eso comenzamos con el comando “git init” para iniciar el proyecto con el control de versiones, una vez iniciado estamos en el “Working Directory”.

Luego de ello podemos proseguir a elaborar nuestro proyecto normalmente, una vez que queramos guardar estos cambios para pasarlos al “Staging área” usamos el comando “git status”, este comando sirve más para controlar el estado de nuestros archivos, nos mostrara un listado de los archivos y que están listos para pasarse a la siguiente área y los que no, los que si estén listos aparecerán listados de color verde, en cambio aquellos que no estén listos o que sufrieron cambios recientemente aparecerán en rojo, para poder pasar preparar esos archivos, luego de ello podemos usar el comando “git add (nombre del archivo)”, esto hará que los archivos seleccionados pasen al “Staging área”, cualquier cambio que haremos se recomienda hacer un git status para saber que archivos habría que preparar para enviar a la siguiente área.

Para poder pasar al siguiente nivel, ósea al “Repository” usamos el comando “git commit” (también es más fácil usar “git commit –m seguido del mensaje para guardar), luego de ello git nos pedirá identificarnos, para que dejar registrado quien realizo cambios en el archivo, para ello seguimos los pasos que nos indica la consola, usamos “git config --global user.email (tu correo)” y “git config --global user.name (tu nombre)” de esa manera quedarías registrado para poder identificar los cambios realizados, con esto quedaría listo para poder crear nuestro repositorio, repetimos el comando “git commit” y se nos abrirá una ventana nueva con los archivos listos para pasar al repositorio, en esta ventana tendremos que ponerle un nombre al repositorio, normalmente se indican los cambios que se realizaron (en caso de escribir y no aparece nada, apretamos “i” para comenzar a escribir) y una vez nombrado presionamos escape y el cursor aparecerá en la parte inferior de la ventana, en ella escribimos “:wq” y presionamos enter, y con esto crearíamos nuestro primer repositorio.

Ahora podemos ver los datos de nuestro repositorio usando el comando “git log” nos dará un “hash” con un numero único que correspondería al commit realizado. Con esto cubrimos los 4 comandos básicos y que más usaremos “git init”, “git add”, “git status” y “git commit”.

Volver en el tiempo… (volver a versiones anteriores)

En el caso de que deseamos volver a una versión anterior o descartar cambios que ya realizamos, usamos el comando “git checkout-- (nombre del archivo en específico)”, esto revertirá los cambios en ese archivo al último commit, esto solo se puede usar antes de que se guarden, ósea usemos git add.

También podemos ver los cambios que se realizan en un archivo en particular, comparándolos con el ultimo commit guardado, para ello usamos el comando “git diff (nombre del archivo especifico)”, nos mostrara en la pantalla de comando con letras verdes las diferencias que se realizaron en este archivo.

Podemos ver la lista de cambios y commit realizados en nuestro archivo usando el comando “git log” con sus nombres y hash correspondientes.

Ignorando archivos.

En el caso de que queramos excluir algún archivo para que no se cargue a nuestro repositorio, debemos hacer lo siguiente; crearemos en nuestra carpeta de proyecto, desde nuestro editor de código, un archivo llamada “. gitignore”, dentro de este, ingresaremos el nombre de los archivos o carpetas que queramos ignorar en nuestro commit

Ramas.

Como ya nombramos, git es una herramienta que se utiliza mucho para hacer proyectos con más de 1 persona, para estos casos, surge la necesidad de dividir el proyecto o simplemente crear versiones nuevas con algunos cambios, para eso usamos las ramas de proyecto, en este caso el proyecto principal será nuestra rama “maestro” y crearemos distintas “ramas” en las cuales realizaremos distintos cambios que podrán o no unirse a la rama maestro, para ello primero debemos crear una rama. En estos casos usaremos el comando “git branch” con esto podremos ver en pantalla la rama maestra y las sub ramas creadas, también nos indicara en que rama estamos ubicados trabajando, en caso de no tener ramas creadas usamos el comando “git branch (nombre de la rama)”, con ello crearemos una nueva rama o espacio de trabajo y para movernos de una rama a otra, usamos “git checkout (nombre de la rama)”.

Una vez que nos encontremos en una rama nueva podremos hacer los cambios correspondientes de manera normal, cada cambio que hagamos se verá reflejado en nuestro git status, ahora si queremos agregar los archivos para pasarlos a nuestra Staging área, debemos usar git add, pero si hemos realizado varios cambios y deseamos pasar todos los archivos, podemos usar el comando “git add .” esto añadirá todo archivo nuevo al siguiente nivel.

Esto solo hará cambios en la rama que estuvimos trabajando, si cambiamos a otra rama, veremos que nuestro código no tiene los cambios que corresponderían a otra rama.

Usando GIT Hub.

Git hub es una página donde podemos almacenar nuestros repositorios de manera remota, desde aquí podemos guardar nuestros proyectos y asignar también a otros desarrolladores para que puedan realizar cambios en el proyecto. Todo archivo se sube a una cuenta que debes crear, puedes publicarlo como público o privado solo para tu cuenta

Para cargar un proyecto en git hub debemos ingresar a nuestra cuenta, y seleccionar la opción de “your repositories” y seleccionamos “new” y le ponemos un nombre a nuestro repositorio donde cargaremos el proyecto, podemos agregarle una descripción y agregar el archivo readme, para finalizar damos a “créate repositorie”. En la ventana que se abrirá tendremos una dirección, que es la de nuestro repositorio para poder cargar el archivo, junto con una lista de pasos para poder cargarlo desde tu pc al repositorio remoto, lo más importante será usar el comando “git remote add origin (dirección del espacio creado en git hub)”, con esto vinculamos nuestro repositorio local con el repositorio virtual en git hub, luego usamos el comando “git push –u origin (rama donde se cargara)” para cargarlo, una vez hagamos esto se nos abrirá una ventana de git hub que nos pedirá logear para poder vincular ambos repositorios.

Clonando nuestro repositorio.

Una vez tengamos ya cargado nuestro repositorio en git hub, podremos clonarlo en cualquier equipo, para ello copiamos la dirección de nuestro repositorio, vamos a la carpeta donde deseamos clonar el repositorio, ejecutamos git para iniciar la ventana de comandos y escribimos “git clone (dirección del repositorio)”.