$git init >> Para inicializar un Proyecto Nuevo

$git status >> Para ver el estado de nuestros archivos

$git add >> Para agregar un archivo al staging area
git add app.js
git add index.html

$git commit >> Para crear un primer punto de control de nuestro Proyecto

$git config -- global user.email >> Para configurar el email de este usuario
git config -- global user.email “pi@gmail.com”

$git config -- global user.name >> Para configurar el nombre del usuario
git config -- global user.name “pie”

Los git config -- global se realizan despues de la instalación de Git. Y se hace antes del git commit aunque no pasa nada si lo ejecutamos, nos pedira que agregemos estas configuraciones y despues de realizarla podemos ejecutar el git commit. Una vez configurado en los siguientes proyectos ya no sera necesario configurarlos.

Cuando modifico los archivos de mi proyecto y los guardo, le doy un git status me sale que el o los archivos han sido modificados. Para ello lo tenemos que agregar a nuestra área de trabajo con git add y luego le damos git commit de esta forma se agregara una nueva versión. Sin embargo de esta forma solo lo estamos guradando de forma lineal, es decir, que solo tenemos un brazo que nos esta guardando todas las modificacions. Lo que quiero decir con esto es que cuando creamos un archivo y se le agrega contenido y le sacamos una foto, a esta misma foto le agregamos mas contenido en base a esta se le añadira y de esta foto se tendra una nueva foto con lo agregado. Mas adelante veremos como se crean mas brasos.

$git checkout -- index.html >> Para revertir los cambios que realizamos anteriormente al archivo. Lo que se quiere decir con esto es que si modificaste el archivo y lo guardaste con este comando se podra revertir ese cambio. Cuando hayamos guardado y le damos git status se mostrara que el archivo fue modificado y con el comando git checkout revierte los cambios y si ahora le damos git status nos saldra que no se a modificado el archivo.

Si modificamos un archivo como hacemos para que los cambios se queden guardados. Una vez modificado lo agregamos al entorno de trabajo con git add index.html. Y lo comiteamos git commit y le agregamos un comentario que describa que fue lo que modificamos

$git diff index.html >> Para ver las diferencias hechas en el archivo. A lo que se refiere es que cuando modificamos un archivo y lo guardamos al realizar este comando se veran en verde las lineas que fueron agregadas y en rojo las que ya tenia. Yo vie que esto se ejecuta bien cuando son archivos de texto plano como por decir .html, .php, etc. sin embargo cuando se le hace a un .docx este no lo soporta y nuesta caracteres ilegibles.

$git commit -m “comentario” >> De esta forma creamos una nueva version del archivo. Al agregar -m ya no nos saldra el editor en donde colocamos el comentario, para eso agregamos -m y ahi mismo hacer el comentario.

Como ignoro archivos o carpetas, se crea un archivo con el nombre .gitignore y ahi escribimos el nombre de la carpeta o archivo que deseemos ignorar. Este archivo se tiene que agregar al entorno de trabajo (git add .gitignore).

Como crear brazos de nuestro proyecto para que asi nuestros colaboradores puedan modificar nuestro proyecto. De esta forma X podra agregar lo que se le pidio y Y tambien podra agregar lo que se pidio, ambos deberan crearse un brazo para poder trabajar por decir X se encarga del login, mientras que Y se encarga de las interfaces.

$git branch >> Muestra el proyecto los brazos que tiene nuestro proyecto. Cuando lo creamos por primera ver nos muestra *master , quiere decir que solo tenemos un brazo y es lineal. Esto pasa cuando lo creamos por primera vez en el futuro podemos crear mas brazos como mencione anteriormente. Sin embargo si creamos brazos para login y interfaces nos parecera de la siguiente forma: *master
login
interfaz
Login y interfaz se crearon de la base de master lo que quiere decir que tienen lo que tiene master sin embargo con la modificaciones que que cada brazo ha realizado el brazo login no tiene lo mismo que tiene el brazo interfaz solo los relaciona la base master que es lo unico en lo que son iguales. A esto se le conoce como una version alternativa

$git branch login >> Con esto creamos una nueva version alternativa. Por ejemplo solo tenemos la version *master, con este comando creamos la version alternativa login. Ahora tenemos 2 versiones. Bien tenemos 2 brazos uno tendra el codigo con login y el otro no (*master).

$git checkout login >> Con este comando cambiamos de version. Por ejempo si ejecutamos git branch nos mostrara: *master
login
Al ejecutar esta comando se cambia de rama, ahora si ejecutamos git branch nos mostrara: master
*login

$git add . >> Agrega ha todos los archivos que hemos creado. Esto nos ayuda a no tener que estar escribiendo a cada momento git add conexion, git add servicio, git add etc. Este los agrega a todos al entorno de trabajo. Y asi luego darle un git commit para crear un nueva foto de nuestro proyecto.