Control de versiones

Es una manera de registrar los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo de tal manera que sea posible recuperar versiones específicas más adelante.
Hubo varias mejoras sobre cómo manejar estas versiones a lo largo del tiempo, llegando a lo que hoy se conoce como **Sistemas de Control de Versiones Distribuidos**.

La idea es darle a cada desarrollador una copia local de todo el proyecto, de esta manera se construye una red **distribuida** de repositorios, en la que cada desarrollador puede trabajar de manera aislada pero teniendo un mecanismo de resolución de conflictos mucho mejor que en versiones anteriores.
Al no existir un repositorio central, cada desarrollador puede trabajar a su propio ritmo, almacenar los cambios a nivel local y mezclar los conflictos que se presenten sólo cuando se requiera. Cómo cada usuario tiene una copia completa del proyecto el riesgo por una caída del servidor, un repositorio dañado o cualquier otro tipo de pérdida de datos es bastante bajo.

En síntesis

1. Podemos volver a cualquier **estado** **anterior** de nuestro proyecto.
2. Podemos tener una historia de cuáles fueron los cambios en el tiempo.
3. Sobre éstos podemos saber **cuando**, **como** y **quien** los realizó.
4. Permite la **colaboración** en un proyecto.
5. Permite desarrollar **versiones** de un mismo proyecto a la vez.

Herramientas

Existen muchos sistemas de control de versiones, como CVS, Subversion, Mercurial y Baazar. En nuestro caso vamos a utilizar **git**

Conceptos fundamentales

repositorio remoto
Es el lugar centralizado donde se guardan los archivos.

repositorio local
Es el lugar dentro de la computadora donde se guardan los archivos.

working directory
Copia del repositorio local, donde voy a empezar a trabajar.

versión
Captura del repositorio en un determinado momento.

commit
Modificaciones que le hacemos a los archivos del repositorio en nuestra computadora.

tag
Es una versión a la que le damos cierta importancia. Ej: 1.0.2

push
Registrar los commits en el repositorio REMOTO.

pull
Obtener los cambios en el repositorio REMOTO.

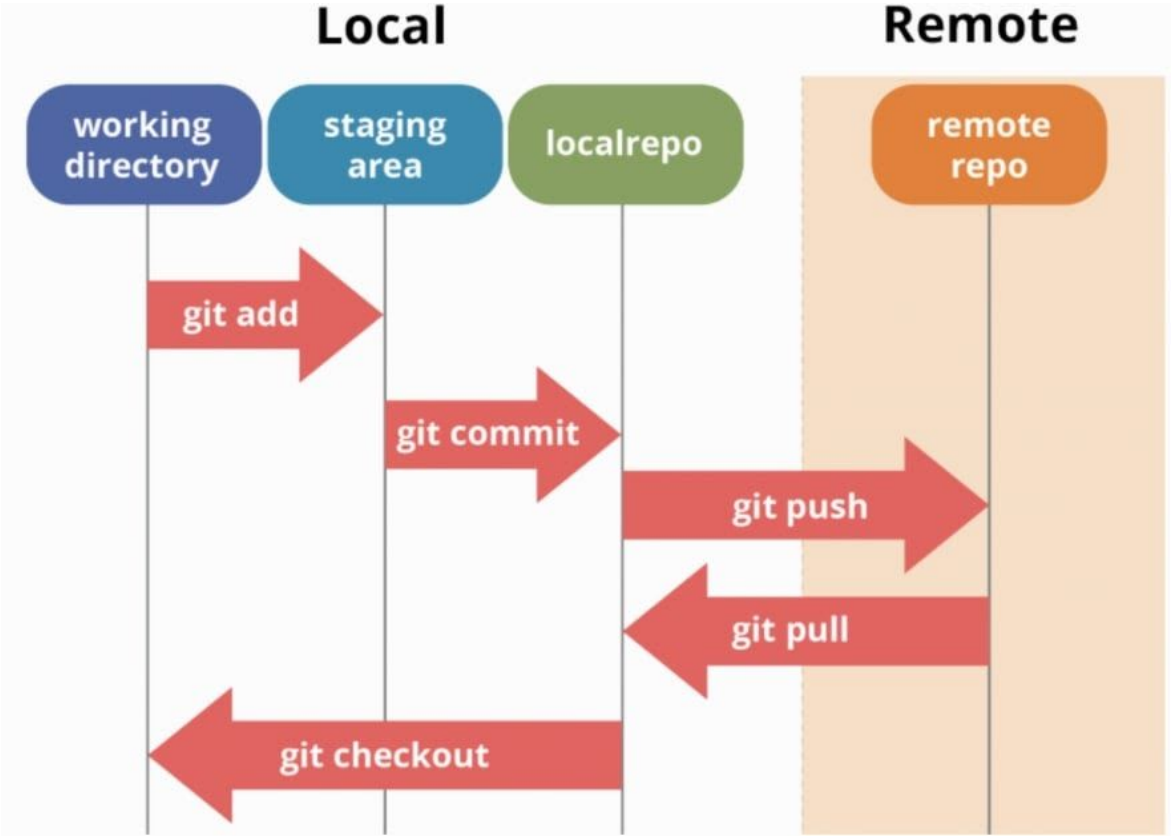
conflicto
Cuando dos o más personas modifican la misma línea de un archivo.

resolución de conflicto
Decidir cuál es la mejor versión que queremos del archivo modificado.

branch
Son referencias a un commit específico o al más actualizado. Por convención tenemos una llamada master.

merge
Realizar una fusión entre dos branch.

Esquema del flujo básico



Comandos básicos

Una vez realizada la instalación de git, la cual es un wizard bastante simple, podemos empezar a conocer los comandos básicos.

```
git init
```

Este comando lo ejecutaremos dentro del directorio que queremos crear como repositorio. Lo que hará, será crear una carpeta llamada .git, con todo lo necesario para el correcto funcionamiento.

Ahora agregamos todos los archivos del proyecto a nuestro repositorio:

```
git add .
```

Ahora vamos a hacer nuestro primer **commit**. Lo que estamos haciendo es guardar todos los cambios realizados, con su correspondiente mensaje. Esto guardará qué cambios se han realizado, quién los ha realizado y el cuando.

```
git commit -m "Se agregan los primeros archivos al proyecto"
```

Una vez guardados todos nuestros cambios, tendríamos que hacer **push** sobre nuestro repositorio. Pero antes, tenemos que definir cuál es nuestro **repositorio remoto**. Nosotros vamos a usar **Github** para la materia. Una vez creado nuestro repositorio, obtendremos una url que apunte a él, y una vez la tengamos ejecutamos la siguiente línea:

```
git remote add origin usuario@yourserver.com:/path/to/proyecto.git
```

Y una vez definido…

```
git push origin master
```

De esta forma, ya tendremos nuestro código almacenado en nuestro servidor GIT, a salvo, y de manera que puedan compartirlo fácilmente con sus compañeros.

En el siguiente enlace tienen una guia muy util para complementar el entendimiento del versionado.