1. Git

Git es un Sistema distribuido de administración de código fuente (SCM).

Sistema de control de versiones

Git, es un software de control de versiones diseñado por Linus Torvalds. Control de versiones es la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo es decir a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o configuración.

Sistema de control de versión son todas las herramientas que nos permiten hacer todas esas modificaciones en nuestro código fuente y hacen que sea más fácil la administración de las distintas versiones de cada producto desarrollado.

Importancia de Git en un proyecto

Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones, cuando estas tienen un gran número de archivos de código fuente, es decir Git nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida.

Algunas características más importantes de Git son:

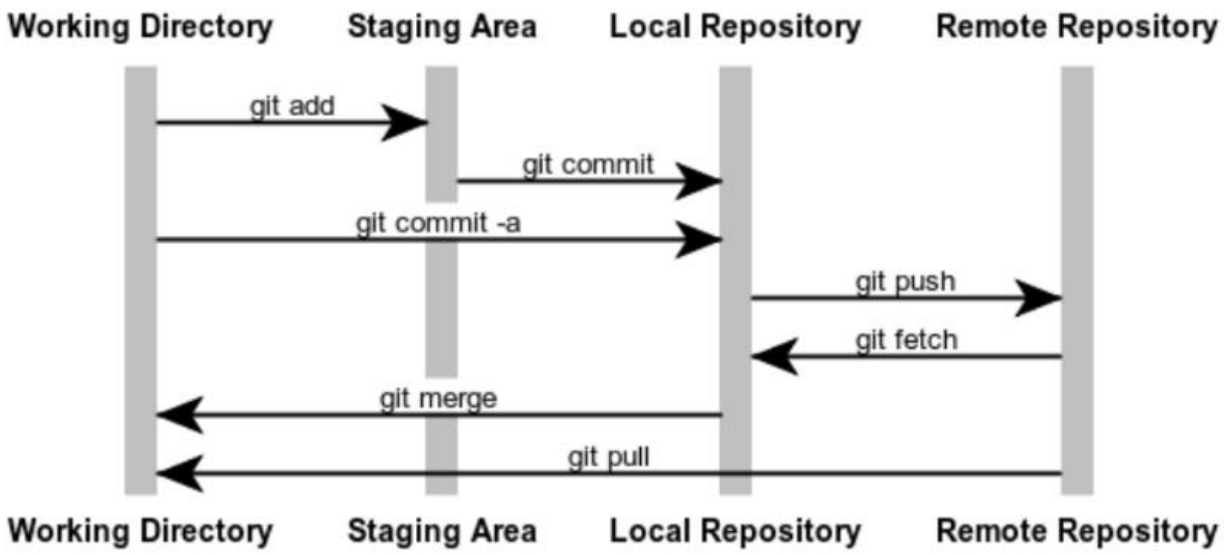
- Rapidez en la gestión de ramas, debido a que Git nos dice que un cambio será fusionado mucho mas frecuentemente de lo que se escribe originalmente.
- Gestión distribuida; los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.
- Gestión eficiente de proyectos grandes.
- Realmacenamiento periódico en paquetes.

1.1. Instalación

Para la instalación de git necesitamos descargar el programa de su página oficial, dependiendo el sistema operativo Windows, Linux y Mac. https://git-scm.com/downloads

1.2. Funcionamiento de git.

- **Working Directory (Directorio de trabajo)**
- **Staging Area (Área de ensayo)**
- **Local Repository (Repositorio local)**
- **Remote Repository (repositorio remoto)**



1.3. Configuración global.

- **git config --global user.name "Nombre usuario"**
- **git config --global user.email "Dirección de correo electrónico"**
- **git config --global color.ui true** **Comando para distinguir las respuestas de git**
- **git config --global --list** **Comando para listar todas configuraciones**

1.4. Crear repositorio.

1.5. Configuración rápida de git.

- **Crear un repositorio con líneas de comando.**

```
echo "# Repositorio-prueba" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/eve0ray/Repositorio-prueba.git
git push -u origin master
```

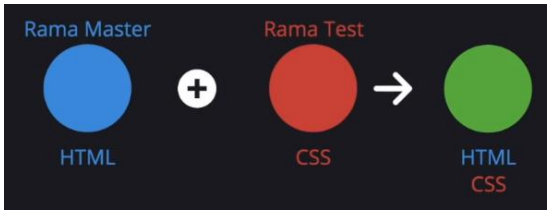
1.6. Cargar y usar git en un proyecto.

- **Dirigirse a la ubicación del proyecto**
- **Git init.-** Este comando marca el inicio de nuestro proyecto, aquí le decimos a Git que empiece a monitorear todos nuestros cambios, solo es necesario hacer esto una ves.
- **Git status.-** Nos muestra el estado de nuestro proyecto.
- **Git add NombreArchivo.-** Comando para pasar los archivos a Staging Area.
- **Git add -A.-** Comando para pasar todos los archivos a Staging Area. Se puede usar "git add . " Añade todos los archivos del directorio.
- **Git commit –m "Mensaje del archivo".-** Guarda los cambios en Local Repository añadiendo un mensaje para identificar los cambios que fueron realizados.
- **Git log**
- **Git checkout código shark.-** Se regresa al commit seleccionado
- **Git checkout master.-** Nos regresa al último commit realizado.
- **Git reset --soft.-** El git reset más simple y que no toca nuestro "working area" (No se mete con nuestro codigo).

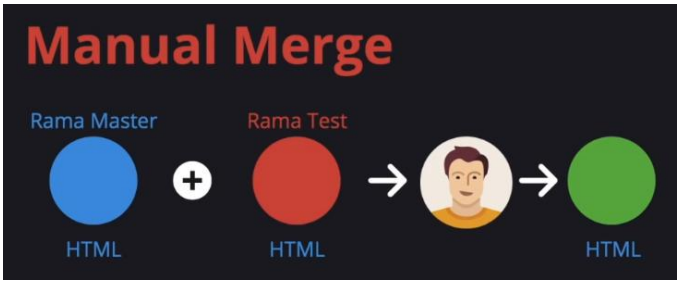
- **Git reset --mixed.-** Este git reset borra el "staging área", sin tocar el "working area".
- **Git reset --hard.-** Este git reset borra absolutamente todo lo que hay en el commit.
- **Git log > nombre.txt.-** Crea un archivo de texto con todos los commits que se tiene.
- **Git help nombreComando.-** Todo la información de los comandos que tiene git.
- **Git remote add**

1.7. Ramas y fusiones en git.

- **Git branch.-** Nos muestra la rama de nuestro proyecto.
- **Git branch nombreRama.-** Crea una rama nueva.
- **Git branch –b nombreRama.-** Crea una rama nueva y se mueve a la rama que se creó.
- **Git checkout nombreRama.-** Nos ayuda a movernos entre ramas.
- **Git branch -D nombreRama.-** Borra una rama.
- **Fusiones.-** Para fusionar dos ramas se debe entrar o situarnos en la rama donde queremos fusionar otra rama, usamos git checkout nombreRama.
- **Git merge nombreRama.-** Comando para fusionar ramas.
- **Fast-Forward.-** Solo va a hacer la fusión, esto pasa normalmente cuando se trabaja con archivos diferentes o líneas de código distintas.



- **Manual Merge.-** Antes de hacer la fusión tiene que pasar por nosotros, normalmente ocurre cuando se trabaja en los mismos archivos o líneas de códigos.



- **Git clone URL.-** Agarra un proyecto de GitHub y lo podemos bajar a nuestro disco local. Para clonar nos copiamos el url = https://github.com/DireccionProyecto
- **Git remote add origin URL.-** Vincula nuestro proyecto local, con nuestro proyecto remoto.
- **Git remote -v.-** Muestra a que repositorio está vinculado nuestro proyecto.