



Xideral

Academia Java

Semana 1 -Dia 3

Configuración y uso de Git.

Presentado por:

Edgar Itzak Sánchez Rogers

Monterrey Nuevo León México a 14 de agosto de 2024

Introducción:

Git es un sistema de control de versiones que permite manejar diferentes versiones de un conjunto de archivos y directorios, puede ser cualquier tipo de archivo y no necesariamente tiene que estar relacionado al mundo de la programación, por ejemplo, pueden ser archivos como hojas de cálculo, archivos de texto, imágenes, etc.

Ramas:

Git permite crear diferentes ramas que representan las diferentes variaciones del proyecto, esto permite trabajar en diferentes ramas de forma simultánea, y posteriormente fusionarlas para una nueva versión, además es posible regresar a versiones anteriores y crear una nueva rama a partir de un punto anterior. La rama principal del proyecto se le conoce como “main” o “master” y es de la cual deriva el resto de ramas.

Comandos básicos de Git:

A continuación, se enlistan comandos básicos de Git y una breve descripción.

Configuración inicial

- `git config --global user.name “[nombre]”` - Configura el nombre de usuario
- `git config --global user.email “[Email]”` - Configura el correo del usuario

Creación de repositorio

- `git init` - crea un repositorio local en la dirección actual
- `git clone [url_repo]` - crea un repositorio de un repositorio ya existente

Información de repositorio

- `git status` - Muestra información del Staging Area
- `git log` - Muestra el historial de commits

Repositorio local

- `git add [archivo]` - Agrega uno o varios archivos de Staging Area
- `git commit` - Crea una nueva instancia de cambios

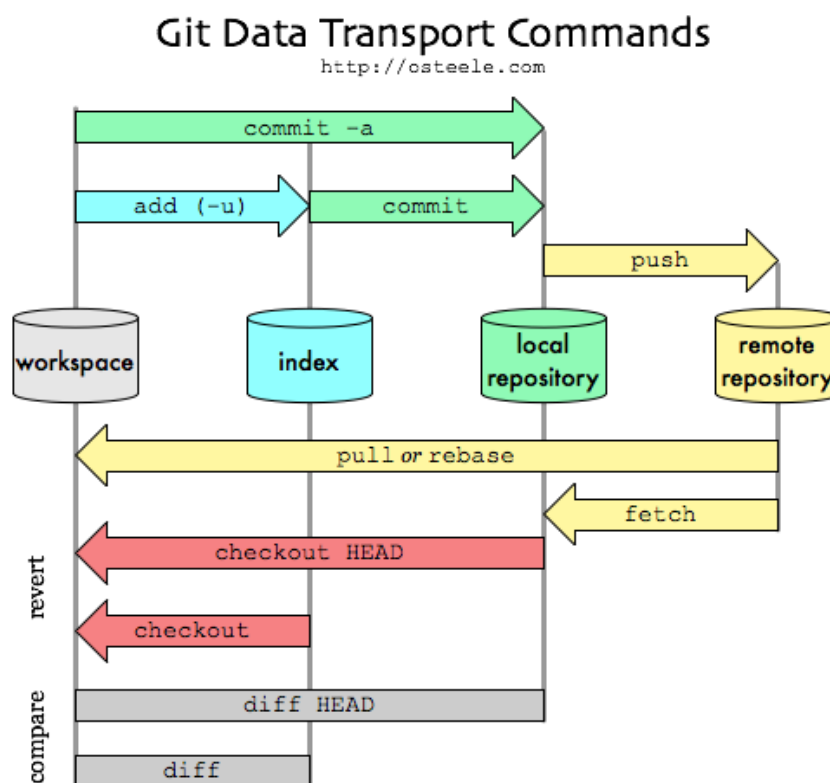
Repositorio remoto

- `git remote add origin [url_repo]` - Conecta con un repositorio remoto
- `git push origin [nombre_rama]` - Envía commits locales al repositorio remoto
- `git pull origin [nombre_rama]` - Fusiona el repositorio local y el remoto

Ramas

- `Git branch [nom]` - Crea una nueva rama
- `Git checkout [nom]` - Cambia de posición a la rama escrita
- `Git merge [nom1][nom2]` - Combina ambas ramas

Diagrama de comandos Git:

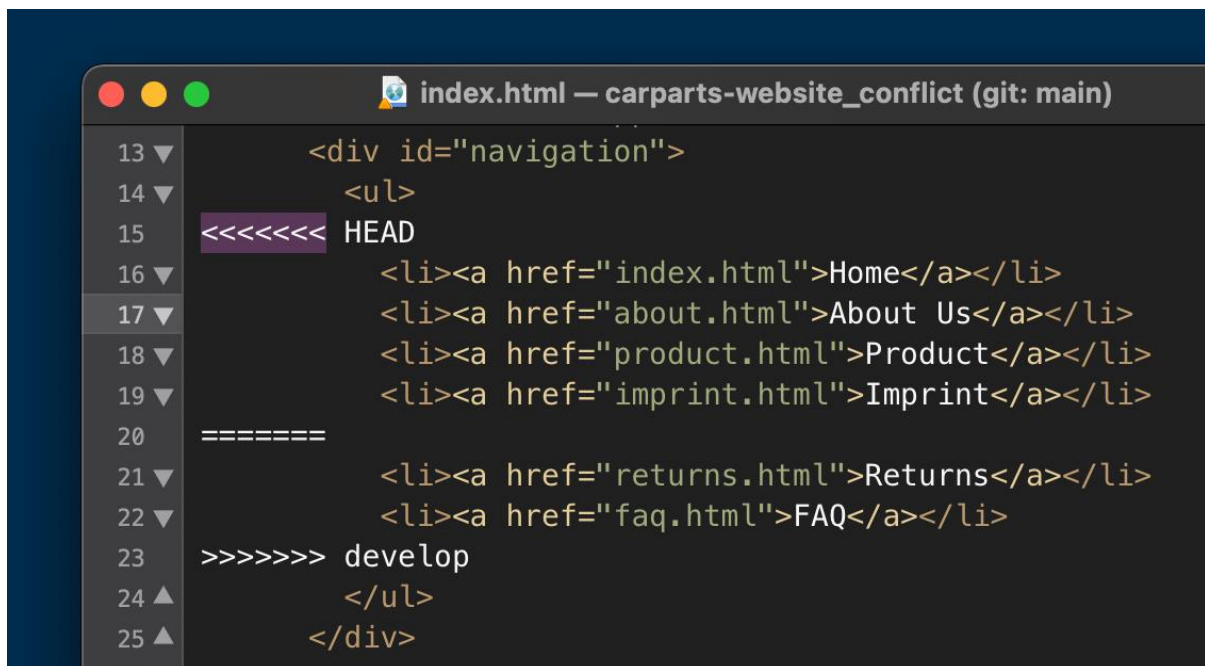


Resolución de conflictos:

Los conflictos de fusión ocurren cuando se intenta fusionar 2 o más ramas en las cuales se haya modificado la misma línea de código. Cuando ocurre un conflicto Git no puede decidir qué línea debería quedarse y cuales descartar. Para solucionar este conflicto es necesario una revisión manual de cada uno de los conflictos y decidir qué cosas mantener, borrar y modificar.

Posteriormente borrar los indicadores:

- <<<<<<<<
- =====
- >>>>>>>>



```
13 <div id="navigation">
14 <ul>
15 <<<<<<< HEAD
16 <li><a href="index.html">Home</a></li>
17 <li><a href="about.html">About Us</a></li>
18 <li><a href="product.html">Product</a></li>
19 <li><a href="imprint.html">Imprint</a></li>
20 =====
21 <li><a href="returns.html">Returns</a></li>
22 <li><a href="faq.html">FAQ</a></li>
23 >>>>>>> develop
24 </ul>
25 </div>
```

Al terminar de resolver todos los conflictos en todos los archivos, se podrán fusionar ambas ramas con normalidad.

Conclusión:

Git es una herramienta de control de versiones que permite a los desarrolladores guardar, documentar y revisar cambios de un código de manera estructurada, esto permite que varias personas puedan trabajar simultáneamente en un mismo proyecto.

Los comandos vistos en este documento son esenciales para trabajar con el control de versiones en proyectos y así gestionar código de manera efectiva, colaborar con otros desarrolladores y llevar un registro de los cambios en los proyectos.

Referencias:

- [1] [Git Commands Ultimate Tutorial \[Part 2\] - DEV Community](#)
- [2] [Git - git-remote Documentation \(git-scm.com\)](#)
- [3] [github - git - remote add origin vs remote set-url origin - Stack Overflow](#)
- [4] [Git Branch | Atlassian Git Tutorial](#)
- [5] [Resolver un conflicto de fusión con la línea de comando - Documentación de GitHub](#)