Material descargable

egg

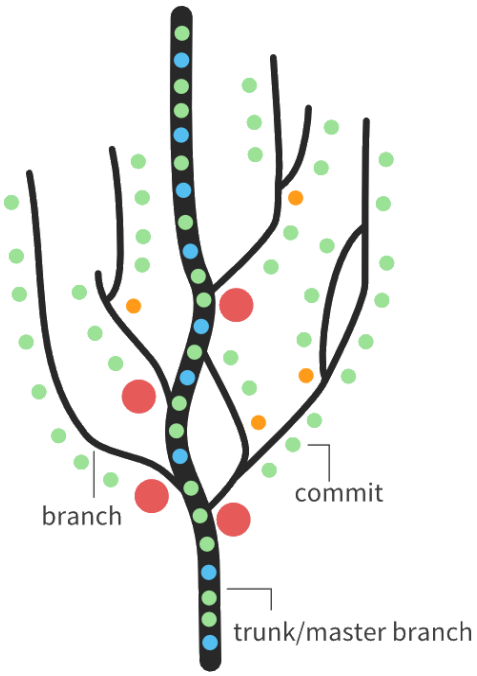
Gestión de Proyectos con Git y GitHub

¿Qué es una rama de Git?

Imagina tu proyecto como un árbol con muchas ramificaciones. La rama principal, comúnmente llamada "main" (o a veces "master"), representa la versión estable de tu proyecto. Las ramas en Git funcionan como "caminos paralelos" donde puedes desarrollar nuevas funcionalidades, experimentar o corregir errores sin alterar el estado de la rama principal.

Cuando creas una rama, estás generando una copia del proyecto en un momento específico. Esto te permite trabajar de manera aislada, realizar pruebas y aplicar cambios sin afectar el proyecto principal. Al finalizar y comprobar que los cambios son correctos, puedes fusionar esta rama con la principal para integrar las mejoras o correcciones.

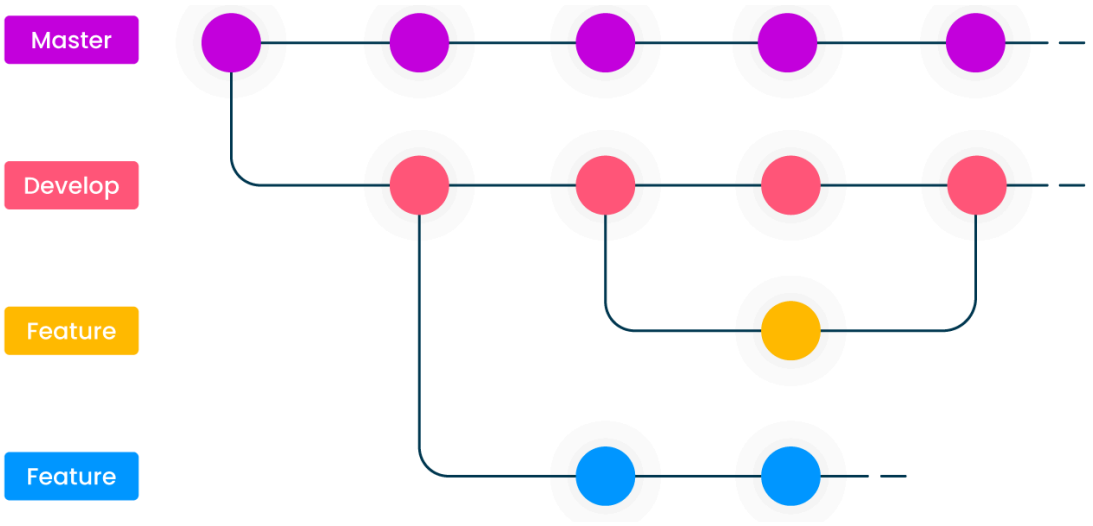
Las ramas también facilitan el trabajo en equipo: cada miembro puede trabajar en una funcionalidad o corrección diferente, y luego integrar su trabajo al proyecto principal de manera ordenada. ¡Es como tener múltiples líneas de tiempo en el desarrollo, todas convergiendo en un mismo proyecto!



¿Qué es el git workflow?

El flujo de trabajo en Git, o Git Workflow, es un conjunto de prácticas recomendadas para trabajar de forma eficiente y organizada en un proyecto. Este flujo define cómo agregar, probar y fusionar nuevas funcionalidades para minimizar conflictos y problemas. Un buen flujo de trabajo en Git debe ofrecer:

- **Facilidad para revertir cambios y corregir errores**: Permite realizar ajustes sin complicaciones si algo sale mal.
- **Escalabilidad**: Se adapta tanto a equipos pequeños como grandes.
- **Claridad y simplicidad**: Facilita la colaboración y asegura que todos trabajen de forma coherente y ordenada.



¿Cómo se planifica y aplica este flujo de trabajo?

Veamos un ejemplo:

Ana y Carlos están trabajando en un proyecto de desarrollo de software. Tienen una rama principal llamada `main`, que contiene la versión estable del proyecto. Para desarrollar nuevas características sin afectar esta versión, ambos crean ramas independientes para trabajar en paralelo:

1. **Asignación de tareas**: Ana se enfocará en crear una nueva funcionalidad de búsqueda, mientras que Carlos trabajará en mejorar la interfaz de usuario.
2. **Creación de ramas**: Ana crea una rama llamada `busqueda` partiendo de `main` en su computadora, mientras que Carlos crea una rama `mejora-interfaz` desde su propia copia de `main`.
3. **Desarrollo en paralelo**: Cada uno trabaja en su rama, realizando commits

regularmente para registrar sus avances. Esto les permite desarrollar y probar sus cambios sin afectar la estabilidad de `main`.

4. **Sincronización con la rama principal**: Periódicamente, Ana y Carlos sincronizan sus ramas con `main`, integrando cualquier cambio reciente para evitar conflictos futuros.
5. **Revisión y fusión**: Una vez completadas sus tareas, ambos suben sus cambios al repositorio remoto. Luego, revisan, corrigen, aprueban e integran sus cambios en `main` para que la versión estable incluya las nuevas funcionalidades y mejoras.
6. **Repetición del ciclo**: Ana y Carlos continúan trabajando en nuevas características, repitiendo este flujo. Gracias a la comunicación constante y a la sincronización regular, ambos logran mantener la estabilidad del proyecto y avanzar en sus tareas de manera coordinada.

Este flujo de trabajo les permite colaborar eficientemente, aprovechando las ramas para trabajar de forma independiente y fusionando cambios solo cuando están listos y revisados

Comandos para Trabajar con Ramas en Git

A continuación, se presentan los comandos esenciales para gestionar ramas en Git, permitiéndote crear, cambiar, borrar y sincronizar ramas en tu repositorio local y remoto:

- **Listar ramas locales :**Muestra todas las ramas locales del repositorio actual.

git branch

- **Crear una nueva rama:** Crea una nueva rama local llamada nombre-de-la-nueva-rama.

git branch nombre-de-la-nueva-rama

- **Borrar una rama local:** Borra la rama local especificada. Ten en cuenta que este comando forzará la eliminación, incluso si la rama contiene cambios sin fusionar.

git branch -D nombre-de-la-rama

- Cambiar de rama: Cambia a la rama especificada en el repositorio local.
  git checkout nombre-de-la-rama

- **Crear y cambiar a una nueva rama al mismo tiempo:** Crea una nueva rama y se cambia a ella inmediatamente.
  git checkout -b nombre-de-la-nueva-rama

- **Cambiar de rama (con `git switch`):** Un comando más reciente y específico para cambiar entre ramas. Funciona de manera similar a `git checkout nombre-de-la-rama`, pero es más intuitivo para este propósito.

  git switch nombre-de-la-rama

- **Subir una rama al repositorio remoto:** Sube la rama especificada al repositorio remoto, lo cual permite compartir la rama con otros colaboradores o hacer un respaldo en el servidor.

  git push origin nombre-de-la-rama

Cada uno de estos comandos es esencial para manejar diferentes líneas de desarrollo en un proyecto, facilitando la organización y colaboración en el flujo de trabajo de Git.

Git tiene una **documentación oficial** completa y detallada. Aquí encontrarás guías y manuales que cubren desde los conceptos básicos hasta temas avanzados, incluyendo configuraciones, comandos y estrategias de flujo de trabajo. La documentación se actualiza con frecuencia para incluir las últimas funcionalidades y mejoras de Git.