Comandos Avanzados de Git

Git es como un cuchillo suizo para los desarrolladores: versátil, poderoso y, a veces, confuso si no sabes qué herramienta usar. Hoy, vamos a explorar algunas de las características y flujos de trabajo más útiles de Git: seleccionar cambios, fusionar con estilo, rebase para una historia más limpia, eliminar esos grandes archivos que accidentalmente comprometiste y deshacer un compromiso cuando te das cuenta de que te has desviado del camino. Vamos a desglosarlo.

Selección de cambios: Tomar solo lo que necesitas Imagina que tienes una rama de características con una docena de compromisos, pero hay un compromiso brillante en ella que quieres extraer y aplicar a tu rama principal, sin traer el resto. Ahí es donde entra `git cherry-pick`.

Es muy sencillo: encuentra el hash del compromiso (puedes obtenerlo de `git log`), cambia a la rama donde lo quieres y ejecuta:

```
git cherry-pick <commit-hash>
```

¡Boom! Ese compromiso ahora es parte de tu rama actual. Si hay un conflicto, Git se detendrá y te dejará resolverlo, igual que una fusión. Una vez que estés satisfecho, compromete los cambios y listo.

Lo uso todo el tiempo cuando una corrección de errores se cuela en una rama de características desordenada y la necesito en `main` lo antes posible. Pero ten cuidado: seleccionar cambios duplica el compromiso, por lo que obtiene un nuevo hash. No esperes que se lleve bien si fusionas la rama original más tarde sin alguna limpieza.

Opciones de fusión: Más que solo "Fusionar" Fusionar es el pan y la mantequilla de Git, pero ¿sabías que viene con sabores? La fusión predeterminada `git merge` hace un "avance rápido"si es posible (enderezando la historia) o crea un compromiso de fusión si las ramas han divergido. Pero tienes opciones:

- `--no-ff` **(Sin avance rápido)**: Fuerza un compromiso de fusión incluso si un avance rápido es posible. Me encanta esto para mantener un registro claro de cuándo una rama de características aterrizó en `main`. Ejecútalo así:

  ```
  git merge --no-ff feature-branch
  ```

- `--squash`: Trae todos los cambios de la rama en un solo compromiso en tu rama actual. Sin compromiso de fusión, solo un paquete único y ordenado. Perfecto para aplastar una rama desordenada en algo presentable:

  ```
  git merge --squash feature-branch
  ```

  Después de esto, necesitarás comprometer manualmente para sellar el trato.

Cada uno tiene su lugar. Tiendo hacia `--no-ff` para ramas de larga duración y `--squash` cuando tengo una rama llena de compromisos "WIP"que prefiero olvidar.

Rebase: Reescribiendo la historia como un profesional Si las fusiones te parecen demasiado desordenadas, `git rebase` podría ser tu vibe. Toma tus compromisos y los reproduce en otra rama, dándote una historia lineal que parece que lo planeaste todo perfectamente desde el principio.

Cambia a tu rama de características y ejecuta:

```
git rebase main
```

Git levantará tus compromisos, actualizará la rama para que coincida con `main` y volverá a colocar tus cambios encima. Si aparecen conflictos, resuélvelos, luego `git rebase --continue` hasta que termine.

La ventaja? Una línea de tiempo impecable. La desventaja? Si ya has empujado esa rama y otros están trabajando en ella, rebase reescribe la historia, ¡prepárate para los correos electrónicos enojados de tus compañeros! Me quedo con rebase para ramas locales o proyectos en solitario. Para cosas compartidas, fusionar es más seguro.

Eliminar grandes archivos de la historia: Oops, ese video de 2GB Todos hemos estado allí: accidentalmente comprometiste un archivo masivo, lo empujaste y ahora tu repo está hinchado. Git no olvida fácilmente, pero puedes limpiar ese archivo de la historia con algo de esfuerzo.

La herramienta para esto es `git filter-branch` o la más nueva `git filter-repo` (recomiendo la última, es más rápida y menos propensa a errores). Digamos que comprometiste `bigfile.zip` y necesitas que se vaya: 1. Instala `git-filter-repo` (consulta su documentación para la configuración). 2. Ejecuta: `git filter-repo --path bigfile.zip --invert-paths` Esto elimina `bigfile.zip` de cada compromiso en la historia. 3. Fuerza el empuje de la historia reescrita: `git push --force`

Atención: esto reescribe la historia, así que coordínate con tu equipo. Y si está en una solicitud de extracción en algún lugar, es posible que también necesites limpiar refs. Una vez que se haya ido, tu repo se reducirá después de una recolección de basura (`git gc`).

Deshacer el compromiso: Revertir el reloj Hiciste un compromiso y lo lamentaste al instante? Git tiene tu espalda. Hay un par de formas de deshacerlo, dependiendo de cuánto te has alejado:

- **Si aún no has empujado**: Usa `git reset`. Para deshacer el último compromiso pero mantener los cambios en tu directorio de trabajo:

  ```
  git reset HEAD^ --soft
  ```

  ¿Quieres deshacerte de los cambios por completo?

  ```
  git reset HEAD^ --hard
  ```

- **Si ya has empujado**: Necesitarás reescribir la historia. Reinicia localmente con `git reset HEAD^`, luego fuerza el empuje:

```
git push --force
```

Nuevamente, esto altera la historia compartida, así que ten cuidado.

Me he salvado con `git reset --soft` más veces de las que puedo contar, perfecto para cuando comprometes demasiado pronto y necesitas ajustar algo.

Conclusión La flexibilidad de Git es lo que lo hace tan poderoso, pero es fácil perderse si no conoces tus opciones. Selecciona cambios para precisión quirúrgica, ajusta las fusiones para que se adapten a tu flujo de trabajo, rebase para una historia pulida y no entres en pánico cuando necesites borrar un error, ya sea un archivo grande o un compromiso apresurado. Practica estos en un repo de prueba si estás nervioso, y pronto se sentirán como segunda naturaleza.

¿Cuál es tu truco favorito de Git? ¡Házmelo saber! Siempre estoy dispuesto a aprender algo nuevo.