Comandos Avanzados en Git

Pull Request

Un Pull Request en Git es una petición para integrar cambios propuestos desde una rama de un repositorio a otra, generalmente de una rama de desarrollo a la rama principal o master. Es una funcionalidad muy utilizada en plataformas de hospedaje de código como GitHub, GitLab o Bitbucket, que facilita la revisión de código, la discusión sobre los cambios propuestos y la colaboración en proyectos de software.

Para qué es un Pull Request

1. Revisión de código: Antes de que los cambios se fusionen en la rama principal, los pull requests permiten que otros colaboradores revisen, discutan y sugieran mejoras en el código propuesto.
2. Colaboración: Fomentan la colaboración entre los miembros del equipo al permitir discusiones en torno a los cambios propuestos.
3. Integración continua: Facilitan la práctica de integración continua al permitir que los cambios se prueben automáticamente antes de su fusión.
4. Control de calidad: Aseguran un control de calidad más riguroso, ya que los cambios deben ser aprobados antes de su integración.

Cómo se usa un Pull Request

El uso de un pull request varía ligeramente entre diferentes plataformas, pero el flujo general es el siguiente:

Paso 1: Crear una rama para los cambios

Antes de trabajar en los cambios que quieres proponer, debes crear una nueva rama en tu repositorio local desde la rama a la que eventualmente querrás fusionar tus cambios (usualmente la rama main o master).

```
git checkout -b <nombre-de-tu-rama>
```

Paso 2: Realizar los cambios

Haz los cambios en tu código en esta nueva rama. Pueden ser correcciones de errores, nuevas características o cualquier otra modificación.

Paso 3: Hacer Commit y Push de tus cambios

Después de realizar los cambios, debes hacer commit a estos cambios en tu rama y luego hacer push de la rama a tu repositorio remoto.

```
git add .
git commit -m "Descripción de los cambios realizados"
git push origin <nombre-de-tu-rama>
```

Paso 4: Crear el Pull Request

Una vez que tus cambios estén en el repositorio remoto, ve a la plataforma que estés utilizando (GitHub, GitLab, Bitbucket) y encontrarás una opción para "Crear un Pull Request" o "New Pull Request". Selecciona la rama que has creado como la fuente de tus cambios y la rama a la que deseas fusionar esos cambios como destino.

Paso 5: Descripción del Pull Request

Debes proporcionar un título descriptivo y una descripción detallada de los cambios propuestos en el pull request. Es útil incluir cualquier contexto relevante, como el problema que resuelven los cambios o cómo se implementaron las soluciones.

Paso 6: Revisión y Discusión

Otros colaboradores del proyecto pueden revisar tus cambios, dejar comentarios, sugerencias o aprobar los cambios. Es posible que necesites hacer ajustes según los comentarios recibidos.

Paso 7: Fusión de los Cambios

Una vez que tu pull request haya sido aprobado y todos los tests automáticos (si los hay) hayan pasado, un responsable del proyecto puede fusionar tus cambios en la rama destino. Esto se puede hacer a través de la interfaz de la plataforma con opciones como "Merge Pull Request" en GitHub.

Paso 8: Limpieza

Después de la fusión, es una buena práctica eliminar la rama de características que utilizaste para los cambios, para mantener limpio el repositorio.

Este flujo permite un control detallado sobre los cambios que se integran en un proyecto, facilitando la gestión de proyectos complejos y la colaboración entre múltiples desarrolladores.

Fork

Un fork es una copia independiente de un repositorio entero. Hacer un fork de un repositorio te permite experimentar con los cambios sin afectar al repositorio original.

Para qué es un Fork

1. Experimentación: Permite a los desarrolladores experimentar con cambios en un proyecto sin afectar el repositorio original.
2. Contribuciones: Facilita a los contribuyentes externos la posibilidad de realizar cambios en su propia copia del repositorio y luego proponer esos cambios al proyecto original a través de pull requests.

3. Proyectos derivados: Si deseas tomar un proyecto en una dirección diferente a la del repositorio original, el fork te permite hacerlo, manteniendo una copia inicial del código para trabajar de manera independiente.
4. Aprendizaje y enseñanza: Es útil para fines educativos, permitiendo a los estudiantes hacer sus propias versiones de un proyecto para aprender y experimentar.

Cómo se usa un Fork

El proceso para hacer y trabajar con un fork es relativamente sencillo y generalmente sigue estos pasos:

Paso 1: Hacer Fork del Repositorio

Ve al repositorio original en la plataforma que estés utilizando (GitHub, GitLab, Bitbucket).

Busca el botón "Fork" generalmente ubicado en la parte superior derecha de la página y haz clic en él.

La plataforma creará una copia del repositorio en tu cuenta personal.

Paso 2: Clonar el Fork a tu Sistema Local

Una vez que el fork está en tu cuenta, puedes clonarlo a tu sistema local para trabajar en él, usando el comando git clone seguido de la URL de tu fork.

```
git clone <url-de-tu-fork>
```

Paso 3: Configurar un Remote Upstream

Para mantener tu fork actualizado con los cambios del repositorio original, es útil configurar un remote adicional llamado upstream que apunte al repositorio original.

```
git remote add upstream <url-del-repositorio-original>
```

Paso 4: Mantener tu Fork Actualizado

Antes de empezar a trabajar en cambios o periódicamente, puedes querer sincronizar tu fork con el repositorio original para obtener los últimos cambios.

```
git fetch upstream
git checkout main  # o la rama principal que estés usando
git merge upstream/main
```

Paso 5: Realizar Cambios y Contribuir

Ahora puedes trabajar en tu fork de manera independiente, hacer cambios, y cuando estés listo para contribuir esos cambios al repositorio original, puedes seguir el proceso de crear una rama, hacer commits, y luego abrir un Pull Request, como se describió anteriormente.

Paso 6: Abrir un Pull Request

Cuando tus cambios estén listos y hayas hecho push de tu rama a tu fork, puedes ir al repositorio original en la plataforma de alojamiento de código y abrir un Pull Request. Tu Pull Request se originará desde tu fork hacia el repositorio original.

Este proceso permite una colaboración abierta y descentralizada en proyectos de software, donde cada persona tiene la libertad de experimentar y proponer cambios sin interferir con el trabajo de los demás.

Rebase

El comando git rebase es una herramienta poderosa de Git utilizada para integrar cambios de una rama a otra, pero de una manera diferente a la fusión (merge). El rebase reescribe la historia de una rama al aplicar sus commits sobre la punta de otra rama, generalmente la principal o base.

Para qué es un Rebase

1. Mantener una Historia Lineal: Ayuda a mantener una historia lineal del proyecto, lo que facilita el seguimiento de los cambios y la comprensión de la historia del proyecto.
2. Simplificar la Revisión de Código: Al presentar los cambios de manera secuencial, el rebase puede hacer más fácil la revisión de código en comparación con el merge, que puede resultar en una historia de proyecto más compleja.
3. Resolver Conflictos: Al aplicar cada commit de la rama de trabajo sobre la base actualizada, se pueden resolver conflictos de forma secuencial, lo que puede simplificar la resolución de conflictos en algunos casos.
4. Preparación para Pull Request: El rebase puede ser usado para actualizar una rama de trabajo con los últimos cambios de la rama principal antes de abrir un Pull Request, asegurando que los cambios sean fáciles de integrar.

Cómo se usa un Rebase

El uso de git rebase implica varios pasos que deben seguirse con cuidado para evitar complicaciones en la historia del repositorio.

Paso 1: Actualizar la Rama Base

Antes de iniciar el rebase, asegúrate de que la rama sobre la que quieres rebasear tus cambios (por ejemplo, main) esté actualizada.

```
git checkout main
git pull origin main
```

Paso 2: Iniciar el Rebase

Cambia a la rama que contiene tus cambios y empieza el rebase con la rama actualizada (en este caso, main).

```
git checkout <tu-rama-de-trabajo>
git rebase main
```

Paso 3: Resolver Conflictos

Si hay conflictos durante el rebase, Git te detendrá y te pedirá que los resuelvas. Puedes hacerlo editando los archivos conflictivos, y luego usar git add para marcarlos como resueltos.

Una vez resueltos todos los conflictos, puedes continuar el rebase con:

```
git rebase --continue
```

Repite este paso hasta que todos los conflictos estén resueltos y el rebase se complete.

Paso 4: Finalizar el Rebase

Una vez que el rebase se completa sin conflictos, los commits de tu rama de trabajo se habrán aplicado sobre la punta de la rama base. Si tu rama de trabajo se ha publicado previamente (push), necesitarás hacer un push forzado para actualizar el repositorio remoto, ya que el rebase cambia la historia de la rama.

```
git push origin <tu-rama-de-trabajo> --force
```

Advertencia: El push forzado puede sobrescribir la historia en el repositorio remoto. Debe usarse con cuidado, especialmente cuando se trabaja en ramas compartidas con otros colaboradores.

Paso 5: Limpieza (Opcional)

Después de que tu rama de trabajo se haya fusionado con la rama principal (por ejemplo, mediante un Pull Request), puedes optar por eliminar la rama de trabajo si ya no la necesitas.

```
git branch -d <tu-rama-de-trabajo>
```

El rebase es una herramienta poderosa, pero debe usarse con cuidado, especialmente en ramas compartidas, ya que cambia la historia del repositorio. Es más seguro usarlo en ramas locales antes de compartir tus cambios con el equipo.