Trabajando con Git: Push, Pull y Merge

Julian F. Latorre

28 de octubre de 2024

Contenido

¿Qué es Git?

- Sistema de control de versiones distribuido
- Permite trabajo simultáneo de múltiples desarrolladores
- Mantiene historial completo de cambios
- Facilita la colaboración y gestión de versiones

Git Push

- Envía cambios locales al repositorio remoto
- Sintaxis básica:

```
git push <remoto> <rama>
```

- Ejemplo:

```
git push origin feature/nueva-funcionalidad
```

Opciones de Git Push

- `git push -u origin <rama>`
  - Establece la rama upstream y hace push
- `git push --all`
  - Envía todas las ramas locales
- `gitpush --tags`
  - Envía todas las etiquetas locales

Buenas prácticas al hacer push

- Haz pull antes de push para evitar conflictos
- Asegúrate de estar en la rama correcta
- Verifica tus cambios con `git status` y `git diff`
- Escribe mensajes de commit claros y descriptivos

Git Pull

- Actualiza el repositorio local con cambios remotos
- Combina `git fetch` y `git merge`
- Sintaxis básica:

```
git pull <remoto> <rama>
```

- Ejemplo:

```
git pull origin main
```

Git Pull vs. Git Fetch

Git Fetch

- Descarga cambios remotos
- No aplica cambios automáticamente
- Permite revisar antes de fusionar

Git Pull

- Descarga cambios remotos
- Fusiona automáticamente
- Más rápido, pero menos control

Opciones de Git Pull

- `git pull --rebase`
  - Aplica commits locales sobre commits remotos
- `git pull --no-commit`
  - Trae cambios sin crear commit de fusión
- `git pull --verbose`
  - Muestra más información durante el proceso

Manejo de conflictos en Git Pull

1. Git marca archivos con conflictos
2. Busca secciones marcadas con `<<<<<<<`, `=======`, `>>>>>>>`
3. Edita manualmente para resolver
4. Usa `git add` para marcar como resuelto
5. Completa el merge con `git commit`

Git Merge

- Combina cambios de diferentes ramas
- Fundamental para integrar trabajo de distintas características
- Dos tipos principales:
  - Fast-forward merge
  - Recursive merge

Sintaxis y Ejemplo de Git Merge

- Sintaxis básica:

```
git merge <rama-a-fusionar>
```

- Ejemplo práctico:

```
git checkout main
git merge feature/nueva-funcionalidad
```

Estrategias de Merge

- Merge commit
  - Crea un nuevo commit que combina las dos ramas
- Squash merge
  - Combina todos los commits de la rama en uno solo
- Rebase
  - Reaplica los commits de tu rama sobre la otra

Resolviendo Conflictos de Merge

1. Identificar archivos con conflictos
2. Editar manualmente para resolver
3. Usar `git add` para marcar como resuelto
4. Completar el merge con `git commit`

Mejores Prácticas para Push, Pull y Merge

- Haz pull frecuentemente
- Trabaja en ramas separadas para cada tarea
- Haz commits pequeños y frecuentes
- Revisa tus cambios antes de push
- Usa pull requests para revisiones de código
- Resuelve conflictos lo antes posible
- Mantén una historia de Git limpia y lineal

Conclusión

- Git push, pull y merge son fundamentales para el trabajo colaborativo
- La práctica regular mejora la eficiencia
- Seguir las mejores prácticas ayuda a evitar problemas comunes
- Dominar estos comandos es esencial para el desarrollo de software moderno