Guía de Git: Comandos y Flujo de Trabajo Básico

Git es un sistema de control de versiones que te permite gestionar y rastrear cambios en tu código. Esta guía te ayudará a entender los comandos básicos y el flujo de trabajo común.

Configuración Inicial

```
# Configurar nombre de usuario
git config --global user.name "Tu Nombre"

# Configurar email
git config --global user.email "tu@email.com"
```

Comandos Básicos

Iniciar un Repositorio

```
# Crear un nuevo repositorio
git init

# Clonar un repositorio existente
git clone <url-del-repositorio>
```

Gestionar Cambios

```
# Ver estado de archivos
git status

# Añadir archivos al área de preparación
git add <archivo>          # Añadir un archivo específico
git add .                  # Añadir todos los archivos modificados

# Confirmar cambios (commit)
git commit -m "Mensaje descriptivo del cambio"

# Ver historial de commits
git log
```

Trabajar con Ramas

```
# Ver ramas existentes
git branch
```

```
# Crear una nueva rama
git branch <nombre-rama>

# Cambiar a otra rama
git checkout <nombre-rama>

# Crear y cambiar a una nueva rama (atajo)
git checkout -b <nombre-rama>

# Fusionar ramas
git merge <nombre-rama>
```

Convención de Nombres para Ramas

Para mantener el repositorio organizado, seguiremos esta convención para nombrar ramas:

- `feature/[iniciales]/[descripcion]`: Para nuevas características
  - Ejemplo: `feature/aa/analisis-exploratorio`
- `hotfix/[iniciales]/[descripcion]`: Para correcciones urgentes
  - Ejemplo: `hotfix/aa/correccion-graficos`
- `docs/[iniciales]/[descripcion]`: Para cambios en documentación
  - Ejemplo: `docs/aa/actualizacion-readme`

Pull Requests (PR)

Un Pull Request (PR) es una solicitud para integrar los cambios de tu rama a la rama principal del proyecto. En el contexto de este curso:

1. **Creación del PR**:
   - Ve a GitHub después de hacer push de tu rama
   - Selecciona tu rama y haz clic en "New Pull Request"
   - Base debe ser `main` y compare tu rama de trabajo
   - Añade un título descriptivo y una descripción detallada de tus cambios
2. **Requisitos del PR**:
   - Título claro que indique el propósito de los cambios
   - Descripción que explique:
     - Qué cambios realizaste
     - Por qué los realizaste
     - Cualquier consideración especial
   - Si aplica, incluye capturas de pantalla o resultados
3. **Proceso de Revisión**:
   - Todos los PRs serán revisados exclusivamente por el instructor (@aladelca)
   - No se permite hacer merge sin la aprobación del instructor
   - Los comentarios de revisión deben ser atendidos con nuevos commits

4. **Flujo de Trabajo con PRs**:

```
# Crear y cambiar a una nueva rama
git checkout -b feature/iniciales/nueva-caracteristica

# Realizar cambios y commits
git add .
git commit -m "Descripción detallada de los cambios"

# Subir la rama a GitHub
git push origin feature/iniciales/nueva-caracteristica
```

5. **Después de la Aprobación**:
    - Solo el instructor puede realizar el merge
    - Una vez merged, puedes eliminar tu rama local:

```
git checkout main
git pull origin main
git branch -d feature/iniciales/nueva-caracteristica
```

Mantenimiento de Ramas

```
# Actualizar tu rama con los últimos cambios de main
git checkout main
git pull origin main
git checkout tu-rama
git merge main

# Eliminar una rama local
git branch -d nombre-rama

# Eliminar una rama remota
git push origin --delete nombre-rama
```

Sincronizar con Repositorio Remoto

```
# Añadir repositorio remoto
git remote add origin <url-repositorio>

# Subir cambios
git push origin <nombre-rama>

# Obtener cambios
```

```
git pull origin <nombre-rama>

# Ver repositorios remotos configurados
git remote -v
```

Flujo de Trabajo Básico

1. **Actualizar tu repositorio local**

   ```
   git pull origin main
   ```

2. **Crear una rama para tu característica**

   ```
   git checkout -b feature/iniciales/nueva-caracteristica
   ```

3. **Realizar cambios y confirmarlos**

   ```
   git add .
   git commit -m "Descripción de los cambios"
   ```

4. **Subir cambios al repositorio remoto**

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

Consejos y Buenas Prácticas

1. **Commits frecuentes**: Realiza commits pequeños y frecuentes con mensajes descriptivos.
2. **Mensajes de commit**: Usa mensajes claros y descriptivos que expliquen el "qué" y el "por qué" de los cambios.
3. **Ramas**:
   - `main` o `master`: Código en producción
   - `feature/`: Para nuevas características
   - `hotfix/`: Para correcciones urgentes
4. **Pull antes de Push**: Siempre haz pull antes de push para evitar conflictos.

Comandos Útiles Adicionales

```

```

```
# Ver diferencias
git diff

# Deshacer cambios en archivos no agregados
git checkout -- <archivo>

# Deshacer último commit (manteniendo cambios)
git reset --soft HEAD~1

# Ver historial de commits de forma gráfica
git log --graph --oneline --all
```

Resolución de Problemas Comunes

Conflictos en Merge

1. Git marcará los archivos con conflictos
2. Abre los archivos y resuelve los conflictos manualmente
3. Después de resolver:

```
git add <archivos-resueltos>
git commit -m "Resueltos conflictos de merge"
```

Cambios en Archivo Equivocado

```
# Deshacer último commit manteniendo cambios
git reset HEAD~1

# Mover cambios a una nueva rama
git checkout -b nueva-rama
git add .
git commit -m "Cambios movidos a nueva rama"
```

Recursos Adicionales

- Documentación oficial de Git
- Git Cheat Sheet
- Learn Git Branching

Recuerda que esta es una guía básica. Git tiene muchas más características y comandos avanzados que puedes explorar según tus necesidades.