Guía Rápida de Comandos Git

Chuleta para Examen

1 Configuración Inicial

1.1 Configurar identidad

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu@email.com"
```

Qué hace: Establece tu nombre y email que aparecerán en todos tus commits.

1.2 Configurar editor

```
git config --global core.editor "nano"
```

Qué hace: Define el editor de texto por defecto para escribir mensajes de commit.

1.3 Activar colores

```
git config --global color.ui true
```

Qué hace: Activa la colorización en la salida de comandos Git para mejor legibilidad.

1.4 Ver configuración

```
git config --list
```

Qué hace: Muestra toda la configuración actual de Git.

2 Operaciones Básicas

2.1 Inicializar repositorio

```
git init
```

Qué hace: Crea un nuevo repositorio Git vacío en el directorio actual.

2.2 Ver estado

```
git status
```

Qué hace: Muestra el estado del directorio de trabajo y el área de staging (archivos modificados, añadidos, eliminados).

2.3 Añadir archivos al staging

```
git add archivo.txt
git add .
```

Qué hace: Añade archivos al área de staging. Con `.` añade todos los cambios.

2.4 Hacer commit

```
git commit -m "Mensaje descriptivo"
```

Qué hace: Guarda una instantánea de los cambios en staging en el historial del repositorio.

2.5 Ver historial

```
git log
git log --oneline
git log --graph --all --oneline
```

Qué hace: Muestra el historial de commits. `--oneline` lo muestra compacto, `--graph` muestra árbol visual.

3 Ver Diferencias

3.1 Diferencias en directorio de trabajo

```
git diff
git diff archivo.txt
```

Qué hace: Muestra cambios no preparados (working directory vs último commit).

3.2 Diferencias en staging

```
git diff --staged
git diff --cached
```

Qué hace: Muestra cambios preparados (staging area vs último commit).

4 Deshacer Cambios

4.1 Descartar cambios en working directory

```
git checkout -- archivo.txt
git restore archivo.txt
```

Qué hace: Descarta cambios no preparados, volviendo al último commit. **¡DESTRUCTIVO!**

4.2 Quitar archivo de staging

```
git reset HEAD archivo.txt
git restore --staged archivo.txt
```

Qué hace: Quita archivo del staging, pero mantiene cambios en working directory.

4.3 Deshacer último commit (mantener cambios)

```
git reset --soft HEAD~1
```

Qué hace: Deshace el último commit, dejando los cambios en staging.

4.4 Deshacer último commit (perder cambios)

```
git reset --hard HEAD~1
```

Qué hace: Deshace el último commit y descarta todos los cambios. **¡MUY DESTRUCTIVO!**

4.5 Enmendar último commit

```
git commit --amend -m "Nuevo mensaje"
```

Qué hace: Modifica el último commit añadiendo nuevos cambios o cambiando el mensaje.

4.6 Revertir commit publicado

```
git revert <commit-hash>
```

Qué hace: Crea un nuevo commit que deshace los cambios de un commit anterior. Seguro para commits compartidos.

4.7 Recuperar archivo eliminado

```
git checkout -- archivo.txt
git restore archivo.txt
```

Qué hace: Restaura un archivo eliminado desde el último commit.

5 Ramas (Branches)

5.1 Ver ramas

```
git branch
git branch -a
```

Qué hace: Lista ramas locales. Con `-a` muestra también remotas.

5.2 Crear rama

```
git branch nombre-rama
```

Qué hace: Crea una nueva rama sin cambiar a ella.

5.3 Cambiar de rama

```
git checkout nombre-rama
git switch nombre-rama
```

Qué hace: Cambia a la rama especificada.

5.4 Crear y cambiar a rama

```
git checkout -b nombre-rama
git switch -c nombre-rama
```

Qué hace: Crea una nueva rama y cambia a ella inmediatamente.

5.5 Eliminar rama

```
git branch -d nombre-rama
git branch -D nombre-rama
```

Qué hace: Elimina una rama. `-d` solo si está fusionada, `-D` fuerza eliminación.

6 Fusiones e Integraciones

6.1 Merge (fusión)

```
git merge nombre-rama
```

Qué hace: Fusiona la rama especificada en la rama actual. Crea un commit de merge si hay divergencia.

6.2 Merge fast-forward

Qué es: Cuando la rama destino no ha avanzado, Git simplemente mueve el puntero hacia adelante sin crear commit de merge.

6.3 Resolver conflictos

1. Editar archivos en conflicto manualmente
2. `git add archivo-resuelto.txt`
3. `git commit` (completa el merge)

Qué hace: Los conflictos aparecen con marcadores «‘‘‘<, =======, ’’’»>. Hay que elegir qué mantener.

6.4 Rebase (reorganización)

```
git rebase nombre-rama
```

Qué hace: Reaplica los commits de la rama actual sobre otra rama, reescribiendo el historial para crear línea recta. **No usar en commits publicados.**

6.5 Continuar rebase tras conflicto

```
git rebase --continue
```

Qué hace: Continúa el rebase después de resolver conflictos.

6.6 Abortar rebase

```
git rebase --abort
```

Qué hace: Cancela el rebase y vuelve al estado anterior.

6.7 Cherry-pick

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

Qué hace: Aplica un commit específico de otra rama a la rama actual, creando un nuevo commit.

7 Archivo .gitignore

7.1 Crear .gitignore

```
echo "*.log" > .gitignore
echo "temp/" >> .gitignore
```

Qué hace: Crea archivo para especificar qué archivos/-carpetas Git debe ignorar. Patrones comunes:

- `*.log` - Todos los archivos .log
- `temp/` - Carpeta temp completa
- `!importante.log` - Excepción, no ignorar

8 Diferencias: Merge vs Rebase

8.1 Merge

Ventajas:

- Preserva historial real
- Seguro para ramas públicas
- Muestra cuando se unieron ramas

Desventajas:

- Historial puede ser complejo
- Crea commits de merge adicionales

8.2 Rebase

Ventajas:

- Historial lineal y limpio
- Más fácil de seguir
- No crea commits de merge

Desventajas:

- Reescribe historial
- **NUNCA** usar en commits publicados
- Puede causar problemas en colaboración

9 Comandos de Ayuda

9.1 Ayuda general

```
git help
git --help
```

9.2 Ayuda de comando específico

```
git help commit
git commit --help
```

10 Conceptos Clave

Working Directory: Directorio con archivos actuales, donde trabajas.

Staging Area: Área intermedia donde preparas cambios antes de commit.

Repository: Base de datos de Git con historial de commits.

HEAD: Puntero al commit actual en la rama activa.

HEAD~1: Commit anterior al actual.

Untracked: Archivo nuevo que Git no rastrea.

Modified: Archivo rastreado con cambios no preparados.

Staged: Archivo preparado para próximo commit.

Fast-forward: Merge sin commit adicional cuando no hay divergencia.

11 Flujo de Trabajo Típico

1. `git status` - Ver estado
2. Modificar archivos
3. `git diff` - Ver cambios
4. `git add .` - Preparar cambios
5. `git commit -m "mensaje"` - Guardar
6. `git log --oneline` - Ver historial

12 Errores Comunes y Soluciones

Hice commit en rama equivocada:

```
git reset --soft HEAD~1
git checkout rama-correcta
git commit -m "mensaje"
```

Añadí archivo equivocado a staging:

```
git restore --staged archivo.txt
```

Necesito deshacer cambios locales:

```
git restore archivo.txt
```

Conflicto en merge:

1. Editar archivos con conflicto
2. `git add archivo-resuelto`
3. `git commit`