Orden	Comando	Herramienta	Tipo	Descripción detallada	Ejemplo
1	**git status**	Git	Información	Muestra el estado del árbol de trabajo y del área de staging: archivos modificados, no rastreados, staged, y la rama actual.	git status
2	git add	Git	Área de staging	Añade archivos al área de staging para incluirlos en el siguiente commit. Puede usarse con rutas, patrones o flags (-A, -p).	git add archivo.txt git add -A # añade todos los cambios
3	git commit	Git	Confirmación	Crea un commit con los cambios que están en el área de staging. Incluye mensaje con -m. Puede enmendarse con --amend.	git commit -m "Mensaje descriptivo" git commit --amend --no-edit # modifica el último commit
4	git push	Git	Remoto	Envía (push) commits locales a un repositorio remoto (por ejemplo origin). -u establece upstream para la rama actual.	git push origin main git push -u origin feature/x
5	git pull	Git	Remoto	Trae y fusiona (fetch + merge) cambios desde el remoto a la rama actual. Frecuente usar --rebase para mantener historial más lineal.	git pull origin main git pull --rebase origin main
6	git clone	Git	Remoto / Inicialización	Clona un repositorio remoto (copia completa con historial) a tu máquina local.	git clone https://github.com/usuario/repositorio.git
7	git branch	Git	Ramas	Lista, crea o borra ramas locales. Con -a muestra ramas remotas. Con -m renombra la rama actual.	git branch # lista ramas locales git branch -a # lista local + remoto git branch -m main # renombrar
8	git switch / git checkout	Git	Ramas / Archivos	Cambia de rama. 'switch' es el comando moderno para cambiar/crear ramas; 'checkout' antiguamente hacía más cosas (restaurar archivos, cambiar ramas).	git switch main git switch -c feature/nueva # con checkout: git checkout -b feature/nueva

9	git merge	Git	Integración	Fusiona otra rama en la rama actual. Soporta opciones como --no-ff para forzar commit de merge.	git checkout main git merge feature/nueva git merge --no-ff feature/nueva
10	git fetch	Git	Remoto	Descarga referencias y objetos desde el remoto pero no altera la rama actual; útil para inspeccionar antes de merge/pull.	git fetch origin git fetch --all --prune # trae todo y limpia referencias remotas borradas
11	git remote	Git	Remoto	Gestiona repositorios remotos: listar (-v), agregar (add), eliminar (remove) y mostrar URL.	git remote -v git remote add origin git@github.com:usuario/repo.git
12	git log	Git	Historial	Muestra el historial de commits. A menudo se usa con --oneline, --graph y --decorate para una vista compacta y visual.	git log --oneline --graph --decorate --all
13	git diff	Git	Comparación	Muestra diferencias entre commits, ramas, o entre working tree y staging. Útil para revisar cambios antes de commit.	git diff # cambios no staged git diff --staged # cambios staged git diff main..feature/nueva
14	git stash	Git	Trabajo temporal	Guarda temporalmente cambios sin commitear para limpiar el working tree y cambiar de rama; luego se recuperan con pop o apply.	git stash save "WIP: prueba" git stash list git stash pop
15	git reset	Git	Historial/Área staging	Reestablece HEAD o archivos: --soft (mantiene cambios en staging), --mixed (por defecto), --hard (descarta cambios). Muy potente y peligroso.	git reset --soft HEAD~1 # deshace último commit dejando cambios staged git reset --hard origin/main # fuerza estado al remoto

16	git restore	Git	Restaurar archivos	Comando moderno para restaurar archivos desde HEAD o desde staging; similar a 'checkout -- <file>' pero más explícito.	git restore archivo.txt git restore --staged archivo.txt # quitar del staging
17	git tag	Git	Marcadores	Crea tags (marcas) en commits: ligeros o anotados (-a). Se usan para versiones/releases. Luego se pueden push con --tags o push origin tag <t>.	git tag -a v1.0 -m "Versión 1.0" git push origin --tags
18	git rebase	Git	Reescritura de historial	Reaplica commits sobre otro punto (ej. base actualizada). Útil para mantener historial lineal; interactivo (git rebase -i) permite editar/ordenar commits.	git fetch origin git rebase origin/main # interactivo: git rebase -i HEAD~5
19	git cherry-pick	Git	Aplicar commit específico	Aplica un commit concreto (por SHA) en la rama actual, útil para traer correcciones puntuales sin hacer merge completo.	git cherry-pick 1a2b3c4d
20	git rm	Git	Eliminar archivos	Quita archivos del working tree y del área de staging (con -f para forzar). Para solo quitar del índice usar --cached.	git rm archivo.txt git rm --cached secreto.txt # quitar del repo pero conservar en disco
21	git mv	Git	Mover/renombrar	Renombra o mueve archivos y registra el cambio para el siguiente commit. Alternativa: mv + git add/remove detecta renombrado también.	git mv viejo.txt nuevo.txt git commit -m "Renombrado"
22	git show	Git	Inspección	Muestra el contenido de un commit, un tag o un objeto: cambios introducidos, mensaje y autor.	git show HEAD git show 1a2b3c4d
23	git bisect	Git	Depuración	Busca de forma binaria el commit que introdujo un bug entre un estado bueno y uno malo; interactivo y muy útil para bugs históricos.	git bisect start git bisect bad # en commit actual git bisect good v1.2.0

24	git blame	Git	Autoría	Muestra línea por línea qué commit y autor introdujo cada línea de un archivo; útil para rastrear la causa de cambios.	git blame src/app.js
25	git reflog	Git	Recuperación	Registro local de movimientos de HEAD. Útil para recuperar commits perdidos o ramas borradas (p. ej. después de reset --hard).	git reflog git checkout HEAD@{5} # volver a un estado anterior
26	git clean	Git	Limpieza	Elimina archivos no rastreados en el working tree. Usar primero con -n (dry-run) para ver qué se borraría.	git clean -n git clean -fd # borra directorios no rastreados
27	git archive	Git	Exportar	Crea un archivo (zip/tar) con el contenido de un commit o rama sin incluir la carpeta .git; útil para distribuciones.	git archive -o proyecto.zip HEAD
28	git submodule	Git	Repositorios anidados	Gestiona submódulos (repositorios dentro de otro). Comandos comunes: add, update, init, sync.	git submodule add https://github.com/otro/lib.git lib/ git submodule update --init --recursive
29	git config	Git	Configuración	Configura opciones globales o locales: user.name, user.email, alias, init.defaultBranch, core.editor, etc.	git config --global user.name "Mark HV" git config --global user.email mhanccova@unsa.edu.pe
30	git rev-parse --show-toplevel	Git	Información	Muestra la ruta del directorio raíz del repositorio (útil para scripts y para confirmar que estás dentro de un repo).	git rev-parse --show-toplevel

31	git remote prune	Git	Mantenimiento remoto	Limpia referencias remotas que ya no existen en el servidor (ej. ramas borradas en remoto).	git remote prune origin
32	git describe	Git	Versionado	Genera una descripción legible del commit usando tags más cercanos (por ejemplo v1.0-2-gabc123).	git describe --tags --long
33	git help / git <comando> --help	Git	Ayuda	Muestra la ayuda integrada y la página del manual para un comando concreto; esencial para aprender opciones y flags.	git help commit git commit --help
34	gh auth login	GitHub (gh CLI)	Autenticación	Inicia sesión en GitHub desde la línea de comandos usando la CLI oficial 'gh'. Puede autenticarse con web o token/SSH.	gh auth login
35	gh repo clone / gh repo create	GitHub (gh CLI)	Repositorios remoto	Clona o crea repositorios directamente con la CLI de GitHub; permite crear y pushear en un solo comando.	gh repo clone usuario/repo gh repo create nombre-repo --public --source=. --push
36	gh pr create / gh pr checkout / gh pr merge	GitHub (gh CLI)	Pull Requests	Gestiona pull requests desde la CLI: crea PRs, cambia a la rama de un PR y mergea PRs (merge, squash, rebase).	gh pr create --base main --head feature/x --title "Feature X" --body "Descripción" gh pr checkout 123 gh pr merge 123 --merge
37	gh issue create / gh issue view	GitHub (gh CLI)	Issues	Crea y consulta issues en repositorios de GitHub desde la CLI; útil para abrir bugs o tareas sin usar el navegador.	gh issue create --title "Bug en login" --body "Pasos para reproducir..."

38	gh release create	GitHub (gh CLI)	Releases	Crea una release en GitHub (tag + notas + binarios opcionales) desde la CLI.	gh release create v1.0 --title "v1.0" --notes "Primera release"
39	gh repo fork	GitHub (gh CLI)	Fork	Crea un fork de un repo remoto en tu cuenta de GitHub y opcionalmente lo clona localmente.	gh repo fork usuario/repo --clone=true
40	gh gist create	GitHub (gh CLI)	Gist	Crea gists (snippets) en GitHub desde la terminal; útil para compartir configuraciones o pequeños scripts.	gh gist create archivo.txt -d "Snippet útil" --public
41	hub (herramienta alternativa)	GitHub (hub CLI)	CLI alternativa	Herramienta histórica similar a 'gh' para integrar GitHub en la terminal. Se menciona por compatibilidad, pero 'gh' es la recomendada actualmente.	hub clone usuario/repo # comandos similares a git
42	Acciones web de GitHub: fork / pull request / issue / stars / watchers	GitHub (Web UI)	Interfaz web	Operaciones típicas en la interfaz de GitHub: forkear repos, abrir PRs, crear issues, marcar stars o ver actions. Muchas organizaciones usan web para revisar y mergear PRs.	En el navegador: botón 'Fork' en un repo; botón 'New pull request' para comparar ramas.
43	git credential-manager / ssh-agent / ssh-keygen	Git/GitHub	Autenticación	Comandos y utilidades para manejar credenciales: crear claves SSH (ssh-keygen), agregar llaves al agente (ssh-add) o usar gestor de credenciales para HTTPS.	ssh-keygen -t ed25519 -C "tu-email@ejemplo.com" ssh-add ~/.ssh/id_ed25519

44	git submodule foreach / update --recursive	Git	Submódulos	Comandos avanzados para ejecutar acciones en todos los submódulos y mantenerlos actualizados de forma recursiva.	git submodule foreach 'git pull origin main' git submodule update --init --recursive
45	git gc / git fsck	Git	Mantenimiento interno	Comandos de limpieza y verificación: git gc optimiza el repositorio; git fsck verifica integridad de objetos. Usarlos con precaución en repos locales grandes.	git gc --aggressive git fsck --full