Git: Quick Command Handbook

Repository

Basic commands to initialize or connect to a remote repository.

create new repo in current directory:

```
git init
```

copy remote repository to local machine:

```
git clone <url>
```

show remote names and URLs (e.g. origin):

```
git remote -v
```

add remote named origin (conventional):

```
git remote add origin <url>
```

Staging & Commits

Stage changes to include them in the next commit, then create a commit with a message.
show working tree and staged changes:

```
git status
```

stage file (use . or -A to add many):

```
git add <file>
```

create a commit from staged changes:

```
git commit -m "msg"
```

unstage file (moves from index back to working tree):

```
git restore --staged <file>
```

Branching & Merging

Branches isolate work; switch between them and combine changes via merge. list local branches:

```
git branch
```

create a branch (does not checkout):

```
git branch <name>
```

switch to branch (older form):

```
git checkout <name>
```

create and switch to a branch (modern form):

```
git switch -c <name>
```

merge <branch> into the current branch:

```
git merge <branch>
```

delete local branch (refuses if unmerged):

```
git branch -d <branch>
```

Remote Workflows

Fetch remote commits, integrate them locally, and push your branches. update remote-tracking branches from origin:

```
git fetch origin
```

fetch and merge (may create a merge commit):

```
git pull
```

fetch and rebase local commits onto remote tip (linear history):

```
git pull --rebase
```

push local branch to origin:

```
git push origin <branch>
```

set upstream so future git push knows where to go:

```
git push -u origin <branch>
```

same as -u:

```
git push --set-upstream origin <branch>
```

safer force push; only if remote hasn't moved unexpectedly:

```
git push --force-with-lease
```

Submodule

Manage nested repositories within a parent repository. add submodule at path:

```
git submodule add <url> <path>
```

initialize submodules (after cloning):

```
git submodule init
```

fetch and checkout submodule commits:

```
git submodule update
```

clone repo with submodules:

```
git clone --recurse-submodules <url>
```

Inspecting History

Examine commit history, changes, and who changed what. compact visual history:

```
git log --oneline --graph --decorate
```

show patches for a file across commits:

```
git log -p <file>
```

show commit details and diff:

```
git show <commit>
```

line-by-line authorship for a file:

```
git blame <file>
```

unstaged changes (working tree vs index):

```
git diff
```

staged changes (index vs HEAD):

```
git diff --staged
```

Undoing Changes

Commands for discarding or moving changes; use with care. discard working changes in <file> (from index/HEAD):

```
git restore <file>
```

restore file content from a specific commit:

```
git restore --source <commit> <file>
```

move HEAD to commit, keep index and working tree (undo commits):

```
git reset --soft <commit>
```

default: move HEAD and reset index, keep working tree:

```
git reset --mixed <commit>
```

reset HEAD, index, and working tree to commit (dangerous):

```
git reset --hard <commit>
```

create a new commit that undoes the specified commit (safe for published history):

```
git revert <commit>
```

Safety Tips

- Prefer git pull –rebase for cleaner history when collaborating.
- Use –force-with-lease instead of –force to avoid overwriting others' work.
- Use annotated tags for releases (git tag -a).
- Run destructive commands (git reset –hard, git push –force) only when sure and ideally on non-shared branches.