Git + Terminal Combined Workflow — Practical Cheat-sheet

This document combines essential terminal commands with a practical Git workflow. Follow the step-by-step sections and copy-paste the commands into your terminal to practice.

Quick Terminal Refresher

pwd

```
Print Working Directory — shows your current directory.

Command:
pwd
```

ls

```
List files/folders in the current directory.

Commands:
ls
ls -l    # detailed list
ls -a    # show hidden files
```

cd

```
Change directory.

Examples:
cd foldername
cd ..   # parent directory
cd ~    # home
cd /path/to/folder  # absolute path
```

mkdir / touch / echo

```
Create folders/files.

Examples:
mkdir projects
touch file.txt
echo "Hello" > file.txt
echo "More" >> file.txt
```

rm / cp / mv

```
Remove, copy, move files/folders.

Examples:
rm file.txt
rm -r folder/
cp src dest
```

```
cp -r folder1 folder2
mv oldname newname
```

. and ..

`.` means current directory; `..` means parent directory.

```
Examples:
git add .
./program.sh
cd ../Documents
```

cat / clear / code .

View file content, clear screen, open VS Code.

```
Examples:
cat file.txt
clear
code .
```

Git Quick Concepts (Plain)

Git

A tool that records snapshots (versions) of your project. Think of it as a version-aware archive.

Repository (repo)

Location where Git stores commits and metadata. Local repo = your machine; remote repo = cloud/host.

Working Directory

Files you actively edit right now.

Staging Area (index)

Where you place files you want included in the next commit (the outbox).

Commit

A saved snapshot plus a message describing changes.

Branch

A movable pointer (bookmark) to a commit. Used to develop features separately.

HEAD

Pointer to the current branch's tip (what you are working on).

Stash

```
Temporary shelf to hide unfinished changes and return to them later.
```

Combined Workflow — Step by Step

1. Create a project folder and initialize Git

```
Terminal:
mkdir myproject
cd myproject

Git:
git init

What it does: creates the .git folder (repository).
```

2. Create files and check status

```
Terminal:
echo "# My Project" > README.md
ls

Git:
git status

What it shows: untracked files (on your desk, not staged).
```

3. Stage and commit (save snapshot)

```
Terminal:
git add README.md
git commit -m "Add README"

Quick:
- git add .  # stage everything
- git commit -m "message"  # save snapshot
```

4. View history and latest commits

```
Commands:
git log --oneline --graph --decorate --all
git log -n 5 --oneline
```

5. Create and switch branches

```
Commands:
git switch -c feature-x  # create and switch
# or: git checkout -b feature-x
git switch main
```

6. Make changes on a branch and merge

```
On feature branch:
echo "New feature" >> feature.txt
git add feature.txt
git commit -m "Add feature"

Merge to main:
git switch main
git merge --no-ff -m "Merge feature-x" feature-x
```

7. Handle conflicts (brief)

```
If conflict occurs during merge, open files with <<<<<<< markers, edit to resolve, then:
git add <file>
git commit

If rebasing: git rebase --continue
```

8. Stash unfinished changes

```
Commands:
git stash push -m "WIP"
# switch branches
git stash list
git stash pop  # restore and remove stash
# or: git stash apply (keep stash)
```

9. Remote basics (push/pull/clone)

```
Commands:
# clone a remote repo
git clone https://example.com/user/repo.git

# add remote (if not cloned):
git remote add origin <url>

# push the main branch:
git push -u origin main

# fetch and merge:
git fetch origin
git pull origin main
```

10. Update local branch safely (fetch + rebase)

```
Commands:
git fetch origin
git rebase origin/main
# resolve conflicts if any, then git rebase --continue

Note: don't rebase public/shared commits.
```

11. Undo safely vs destructively

```
Safe (preserves history):
```

```
git revert <commit>

Destructive (rewrites history):
git reset --hard <commit>

Exam tip: prefer revert for published commits.
```

12. Recover lost commits

```
Command:
git reflog
# find the lost HEAD and reset:
git reset --hard <reflog-hash>
```

Useful Command Cheatsheet

Status & diff

```
git status
git diff
git diff --staged
```

Staging & commit

```
git add <file>
git add .
git commit -m "msg"
```

Branches

```
git branch
git switch <branch>
git switch -c <branch>
git branch -d <branch>
```

Merge & Rebase

```
git merge <branch>
git merge --no-ff -m "msg" <branch>
git rebase <base>
```

Remote

```
git remote -v
git fetch
git pull
git push
git clone <url>
```

Tags & cherry-pick

```
git tag -a v1.0 -m "msg"
```

```
git push --tags
git cherry-pick <start>^..<end>
```

Logs & show

```
git log --oneline --graph --all
git show <commit>
```

Undo

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