Git Power User Cheatsheet

50+ commands, workflows & tricks the docs don't teach you

Undo & Recovery

Undo last commit (keep changes)

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

Undo last commit (discard changes)

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

Recover deleted branch

```
git reflog | grep 'branch-name'
git checkout -b branch-name HEAD@{n}
```

Undo a pushed commit safely

```
git revert
```

Restore a single file from commit

```
git checkout -- path/to/file
```

Find lost commits

```
git fsck --lost-found
```

Undo a git add

```
git reset HEAD
```

Unstage everything

```
git reset HEAD .
```

Interactive Rebase Mastery

Squash last 5 commits

```
git rebase -i HEAD~5
# Change 'pick' to 'squash' on lines 2-5
```

Reorder commits

```
git rebase -i HEAD~n
# Move lines to reorder
```

Edit a commit message

```
git rebase -i HEAD~n
# Change 'pick' to 'reword'
```

Split a commit

```
git rebase -i HEAD~n
# Mark as 'edit', then:
git reset HEAD~1
git add -p
git commit
git rebase --continue
```

Rebase onto specific branch

```
git rebase --onto target start end
```

Search & Forensics

Find who wrote a line

```
git blame -L 10,20 file.py
```

Search all history for string

```
git log -S 'search_term' --all
```

Search commit messages

```
git log --grep='pattern'
```

Find when a bug was introduced

```
git bisect start
git bisect bad HEAD
git bisect good v1.0
# Git binary-searches for you
```

Show file at specific commit

```
git show commit:path/to/file
```

List files changed between branches

```
git diff --name-only main..feature
```

Find large files in history

```
git rev-list --objects --all | \
git cat-file --batch-check | \
sort -k3 -n -r | head -20
```

Stash Like a Pro

Stash with a name

```
git stash push -m 'WIP: feature X'
```

Stash specific files

```
git stash push -m 'partial' -- file1 file2
```

Stash untracked files too

```
git stash -u
```

Apply specific stash

```
git stash apply stash@{2}
```

Create branch from stash

```
git stash branch new-branch stash@{0}
```

Show stash diff

```
git stash show -p stash@{0}
```

Advanced Workflows

Cherry-pick without committing

```
git cherry-pick --no-commit
```

Create patch file

```
git format-patch -1
```

Apply patch

```
git am < patch-file.patch
```

Worktrees (multiple checkouts)

```
git worktree add ../feature-branch feature
```

Sparse checkout (huge repos)

```
git sparse-checkout init --cone
git sparse-checkout set src/module
```

Shallow clone (save bandwidth)

```
git clone --depth=1 --branch=main
```

Track upstream changes

```
git remote add upstream
git fetch upstream
git merge upstream/main
```

Aliases That Save Hours

Pretty log graph

```
git config --global alias.lg \
"log --graph --oneline --decorate --all"
```

Show last commit

```
git config --global alias.last \
"log -1 HEAD --stat"
```

Undo last commit

```
git config --global alias.undo \
"reset --soft HEAD~1"
```

List branches by date

```
git config --global alias.recent \
"branch --sort=-committerdate --format='%(committerdate:short) %(refname:short)'"
```

Quick amend (no edit)

```
git config --global alias.oops \
"commit --amend --no-edit"
```

.gitconfig Power Settings

Auto-correct typos

```
[help]
autocorrect = immediate
```

Better diff algorithm

```
[diff]
algorithm = histogram
```

Reuse recorded resolutions

```
[rerere]
enabled = true
```

Sign commits with SSH

```
[gpg]
format = ssh
[user]
signingkey = ~/.ssh/id_ed25519
```

Default branch = main

```
[init]
defaultBranch = main
```

Push current branch only

```
[push]
default = current
```

■ Pro Tips

Use **git reflog** before panicking — it records everything for 90 days.

Set **rerere.enabled = true** to auto-resolve repeated merge conflicts.

Use **git worktree** instead of stashing when switching contexts frequently.

Add **git maintenance start** in large repos for background optimization.

Use **--autostash** with rebase to avoid manual stash/pop cycles.