GIT

CHEATSHEET

Every command explained — Setup · Clone · Branch · Merge · Rebase · Remote · Stash · Tag · Bisect · and more

140 commands · 14 categories

@contentintech

contentintech.com

▸ SETUP & CONFIG

$ git config --global user.name "Name"
Set your global commit author name

$ git config --global user.email "e@x"
Set your global commit author email

$ git config --global core.editor vim
Set default editor for commit messages

$ git config --list
Show all config values in effect

$ git config --global alias.st status
Create a shortcut alias (e.g. git st)

$ git config --global core.autocrlf input
Handle line-ending conversions

▸ INIT & CLONE

$ git init
Initialize a new local repository

$ git init <dir>
Create a new repo inside <dir>

$ git clone <url>
Clone a remote repo into a new folder

$ git clone <url> <dir>
Clone into a specific local directory

$ git clone --depth 1 <url>
Shallow clone — only latest snapshot

$ git clone --branch <b> <url>
Clone and check out a specific branch

▸ STAGING & SNAPSHOT

$ git status
Show working-tree and staging status

$ git status -s
Short/compact status output

$ git add <file>
Stage a specific file

$ git add .
Stage all changes in current directory

$ git add -p
Interactively stage hunks (partial adds)

$ git diff
Show unstaged changes

$ git diff --staged
Show staged changes (vs last commit)

$ git diff <b1>..<b2>
Diff between two branches or commits

$ git commit -m "msg"
Commit staged changes with a message

$ git commit -am "msg"
Stage tracked files and commit in one step

$ git commit --amend
Rewrite the last commit (msg or content)

$ git rm <file>
Remove file and stage the deletion

$ git rm --cached <file>
Untrack file but keep it on disk

$ git mv <old> <new>
Rename/move a file and stage the change

▸ BRANCHING

$ git branch
List all local branches

$ git branch -a
List local and remote branches

$ git branch <name>
Create a new branch at current HEAD

$ git branch -d <name>
Delete a branch (safe — merged only)

$ git branch -D <name>
Force-delete a branch (unmerged OK)

$ git branch -m <old> <new>
Rename a branch

$ git branch --merged
Show branches merged into current

$ git switch <name>
Switch to an existing branch

$ git switch -c <name>
Create and switch to a new branch

$ git checkout <name>
Switch branch (classic syntax)

$ git checkout -b <name>
Create and switch (classic syntax)

$ git checkout -b <n> origin/<n>
Track a remote branch locally

▸ MERGING & REBASING

$ git merge <branch>
Merge <branch> into current branch

$ git merge --no-ff <branch>
Merge and always create a merge commit

$ git merge --squash <branch>
Squash branch into one staged diff

$ git merge --abort
Abort an in-progress merge

$ git rebase <branch>
Rebase current branch onto <branch>

$ git rebase -i HEAD~N
Interactive rebase on last N commits

$ git rebase --continue
Continue after resolving a conflict

$ git rebase --abort
Cancel an in-progress rebase

$ git rebase --skip
Skip the conflicting commit and continue

$ git cherry-pick <sha>
Apply a specific commit onto current branch

$ git cherry-pick <s1>..<s2>
Apply a range of commits

▸ REMOTE REPOS

$ git remote -v
List remote connections with URLs

$ git remote add <name> <url>
Add a new remote

$ git remote remove <name>
Remove a remote

$ git remote rename <old> <new>
Rename a remote

$ git remote set-url <n> <url>
Change URL of an existing remote

$ git fetch <remote>
Download remote changes without merging

$ git fetch --all
Fetch all remotes

$ git pull
Fetch and merge the tracked remote branch

$ git pull --rebase
Fetch and rebase instead of merge

$ git push <remote> <branch>
Push branch to remote

$ git push -u origin <branch>
Push and set upstream tracking

$ git push --force-with-lease
Safe force-push (fails if remote changed)

$ git push origin --delete <branch>
Delete a remote branch

$ git push --tags
Push all local tags to remote

▸ STASHING

$ git stash
Save dirty working tree to stash stack

$ git stash push -m "msg"
Stash with a descriptive message

$ git stash list
Show all stash entries

$ git stash pop
Apply top stash and remove it

$ git stash apply stash@{N}
Apply a specific stash without removing

$ git stash drop stash@{N}
Delete a specific stash entry

$ git stash clear
Delete all stash entries

$ git stash branch <name>
Create a branch from top stash

$ git stash show -p
Show diff of the top stash

▸ INSPECTING & LOGGING

$ git log
Show full commit history

$ git log --oneline
Compact one-line-per-commit view

$ git log --oneline --graph --all
Visual ASCII branch graph

$ git log -p
Show patch/diff for each commit

$ git log --author="Name"
Filter log by author name

$ git log --since="2 weeks ago"
Filter log by time range

$ git log -- <file>
History of a specific file

$ git log <b1>..<b2>
Commits in b2 not in b1

$ git show <sha>
Show details and diff of a commit

$ git show <sha>:<file>
Show file content at a commit

$ git blame <file>
Show who changed each line and when

$ git shortlog -sn
Commit count per author, sorted

$ git reflog
Log of every HEAD movement (recovery)

▸ UNDOING CHANGES

$ git restore <file>
Discard working-tree changes in a file

$ git restore --staged <file>
Unstage a file (keep changes)

$ git reset HEAD <file>
Unstage a file (classic syntax)

$ git reset --soft HEAD~1
Undo last commit, keep changes staged

$ git reset --mixed HEAD~1
Undo last commit, unstage changes

$ git reset --hard HEAD~1
Undo last commit, discard all changes

$ git revert <sha>
Create a new commit undoing <sha>

$ git revert --no-commit <sha>
Stage the revert without committing

$ git clean -fd
Delete untracked files and directories

$ git clean -n
Dry-run: show what clean would remove

▸ TAGGING

$ git tag
List all tags

$ git tag <name>
Create a lightweight tag at HEAD

$ git tag -a <name> -m "msg"
Create an annotated tag with message

$ git tag -a <name> <sha>
Tag a specific past commit

$ git tag -d <name>
Delete a local tag

$ git push origin <tag>
Push a single tag to remote

$ git push origin --delete <tag>
Delete a remote tag

$ git describe --tags
Show nearest tag + distance + sha

▸ SEARCHING

$ git grep "pattern"
Search working tree for a regex pattern

$ git grep -n "pattern"
Show line numbers in grep results

$ git log -S "string"
Find commits that added/removed string

$ git log -G "regex"
Find commits whose diff matches regex

$ git log --all --full-history -- f
Find all history of a deleted file

$ git bisect start
Begin binary search for a bad commit

$ git bisect bad
Mark current commit as broken

$ git bisect good <sha>
Mark a known-good commit

$ git bisect reset
End bisect session, restore HEAD

▸ SUBMODULES

$ git submodule add <url> <path>
Add a repo as a submodule

```
$ git submodule init
  Initialize submodule config entries

$ git submodule update
  Fetch and checkout submodules

$ git submodule update --init --recursive
  Init + update all, recursively

$ git submodule foreach git pull
  Pull latest in every submodule

$ git submodule status
  Show SHA and state of each submodule

$ git submodule deinit <path>
  Unregister a submodule
```

▸ WORKTREES & PATCHES

```
$ git worktree add <path> <branch>
  Check out a branch in a new directory

$ git worktree list
  List all linked worktrees

$ git worktree remove <path>
  Remove a linked worktree

$ git format-patch HEAD~N
  Export last N commits as .patch files

$ git am <file.patch>
  Apply a patch file as a commit

$ git apply <file.patch>
  Apply patch to working tree only

$ git bundle create out.bundle --all
  Bundle entire repo to a single file

$ git bundle unbundle out.bundle
  Restore from a bundle file
```

▸ ADVANCED & MAINTENANCE

```
$ git archive --format=zip HEAD > o.zip
  Export HEAD as a zip archive

$ git gc
  Run garbage collection on the repo

$ git fsck
  Verify integrity of object database

$ git prune
  Remove unreachable loose objects

$ git count-objects -vH
  Show disk usage of repo objects

$ git cat-file -p <sha>
  Print raw content of any git object

$ git rev-parse HEAD
  Print the full SHA of HEAD

$ git rev-parse --abbrev-ref HEAD
  Print current branch name

$ git ls-files
  List files tracked by git

$ git ls-remote <remote>
  List references in a remote repo

$ git for-each-ref --sort=-date ...
  Query refs with custom format

$ git notes add -m "msg" <sha>
  Attach a note to a commit

$ git replace <old-sha> <new-sha>
  Transparently replace one object
```

STAY CONNECTED

Follow us for daily tech content, tutorials, and resources

INSTAGRAM

@contentintech

instagram.com/contentintech

YOUTUBE

@contenintech

youtube.com/@contenintech

WHATSAPP

Contentintech Channel

whatsapp.com/channel/0029VbCZJQU05MUijd2ofT1D

Thanks for learning with us — share this cheatsheet with your team!

contentintech.com