Git + GitHub Actions Cheat Sheet

devtoolhub.com
Tested guides for developers

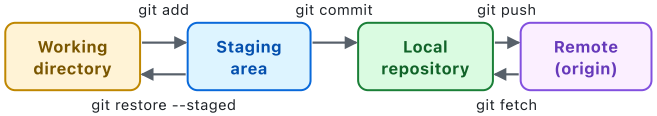
Undo things (what most people search for)

Discard unstaged edits in a file	`git restore <file>`
Unstage a file, keep the edits	`git restore --staged <file>`
Fix the last commit message	`git commit --amend` Only if not pushed
Add a forgotten file to last commit	`git add <file> && git commit --amend --no-edit` Only if not pushed
Undo last commit, keep changes staged	`git reset --soft HEAD~1`
Undo last commit, keep changes unstaged	`git reset HEAD~1` Same as --mixed (the default)
Throw the last commit away	`git reset --hard HEAD~1` **Destructive: uncommitted work is lost**
Undo a pushed commit safely	`git revert <sha>` Adds a new commit that reverses it
Recover a "lost" commit	`git reflog` Then git checkout <sha>; entries expire after ~90 days

Branching

```
git switch -c feature/x          # create + switch
git switch main                  # switch to a branch
git branch -d feature/x          # delete (merged only)
git branch -D feature/x          # force-delete
git push -u origin feature/x     # first push, set upstream
git push origin --delete feature/x   # delete on remote
git fetch --prune                # drop refs deleted remotely
```

Where your work lives



git restore <file> discard working-directory edits (staging stays)
git pull = git fetch + git merge (git pull --rebase = fetch + rebase)
git stash shelves edits from the first two boxes git stash pop brings them back
git switch <branch> moves HEAD and updates the working directory

Merge conflicts

```
git merge main            # or: git rebase main
git status                # conflicted files show as UU
# edit files, remove the <<<<<<< ======= >>>>>>> markers
git add <resolved-file>
git merge --continue      # or: git rebase --continue
git merge --abort         # back to before the merge
git rebase --abort        # back to before the rebase
```

Set a merge tool once: `git config --global merge.tool vimdiff` (or `meld`, `vscode`)

Rebase or merge?

- **Merge** into shared branches (`main`): keeps history as it happened.
- **Rebase** your own branch onto `main` before a PR: linear history.
- **Never** rebase commits other people have pulled.
- `git pull --rebase` avoids noise merge commits on sync.

Stash

```
git stash push -m "wip: auth refactor"
git stash list
git stash pop               # apply latest + drop it
git stash apply stash@{2}   # apply, keep it
git stash drop stash@{2}
```

Tags & releases

```
git tag -a v1.4.0 -m "Release 1.4.0"   # for releases
git push origin v1.4.0
git push origin --tags                  # push all tags
git tag -d v1.4.0                       # delete locally
git push origin :refs/tags/v1.4.0       # delete remote
```

Cherry-pick & clean

```
git cherry-pick <sha>           # apply one commit
git cherry-pick <sha1>^..<sha2> # a range, incl. sha1
git cherry-pick --abort         # bail out
git clean -n                    # preview deletions
git clean -fd                   # delete untracked
git clean -fdx                  # ...and ignored too
```

Inspect

```
git log --oneline --graph --decorate --all
git log -p <file>        # file history with diffs
git blame <file>
git show <sha>
git diff main...feature/x    # what the branch adds
```

Which undo do I need? Ask: has it left my machine?

Not committed yet
Use `git restore <file>` to discard edits, or `git restore --staged <file>` to unstage.

Committed, not pushed
Use `git commit --amend` for the last commit, or `git reset --soft HEAD~1` to redo it.

Already pushed and shared
Use `git revert <sha>`. It adds a new commit, so nobody's history is rewritten.

Minimal workflow

```
# .github/workflows/ci.yml
name: CI
on:
  push:
    branches: [main]
  pull_request:

permissions:
  contents: read

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v7
      - uses: actions/setup-node@v7
        with:
          node-version: 26
          cache: npm
      - run: npm ci
      - run: npm test
```

Latest majors checked Nov 2026: `checkout@v7`, `setup-node@v7`, `cache@v6`. Node 26 is Active LTS (since 28 Oct 2026); 24 and 22 are Maintenance; 20 reached end of life.

Matrix builds

```
strategy:
  fail-fast: false
  matrix:
    os: [ubuntu-latest, macos-latest]
    node: [22, 24, 26]
runs-on: ${{ matrix.os }}
```

Caching

```
- uses: actions/cache@v6
  with:
    path: ~/.cache/pip
    key: ${{ runner.os }}-pip-${{ hashFiles('requirements.txt') }}
    restore-keys: ${{ runner.os }}-pip-
```

Entries unused for 7 days are evicted; default cap is 10 GB per repository.

Triggers (the on: key)

Commits pushed	`push` Filter with branches, paths, tags
Pull request opened or updated	`pull_request`
Manual “Run workflow” button	`workflow_dispatch` Define inputs
On a schedule	`schedule` cron: '0 6 * * 1' (UTC only)
Called by another workflow	`workflow_call`
Release published	`release`
PR, but with secrets	`pull_request_target` Runs in the base repo. Never run untrusted PR code here

Path & branch filters

```
on:
  push:
    branches: ['main', 'release/**']
    paths: ['src/**', '!**.md']
```

Permissions: least privilege

```
permissions:
  contents: read        # default to read-only
# raise per job only where needed:
#   contents: write     # push tags/commits
#   packages: write     # publish to GHCR
#   id-token: write     # OIDC cloud login
```

Secrets & variables

```
steps:
  - run: ./deploy.sh
    env:
      API_TOKEN: ${{ secrets.API_TOKEN }}
      REGION: ${{ vars.AWS_REGION }}
```

- Secrets are masked in logs; variables are not.
- `GITHUB_TOKEN` is automatic: scope it with `permissions:`.
- Prefer **OIDC** over long-lived cloud keys.

Security quick rules

1. Pin third-party actions to a **full commit SHA**, not a tag. Tags can be moved.
2. Default to `permissions: contents: read`; raise per job.
3. Never check out and run untrusted PR code under `pull_request_target`.
4. Use OIDC for cloud auth, not stored keys.
5. Treat `github.event.*` text (PR titles, branch names) as untrusted: never paste it straight into `run:`.

Why rule 1: in March 2025 every tag of `tj-actions/changed-files` was repointed to a malicious commit that leaked CI secrets (CVE-2025-30066). A pinned SHA cannot be moved.

Concurrency: cancel old runs

```
concurrency:
  group: ${{ github.workflow }}-${{ github.ref }}
  cancel-in-progress: true
```

Reusable workflow (caller)

```
jobs:
  deploy:
    uses: org/repo/.github/workflows/deploy.yml@main
    with: { environment: prod }
    secrets: inherit
```

Expressions & contexts

Commit SHA	`${{ github.sha }}`
Branch or tag name	`${{ github.ref_name }}`
What triggered it	`${{ github.event_name }}`
Who triggered it	`${{ github.actor }}`
Runner OS	`${{ runner.os }}`
Job result	`${{ job.status }}`
Only on main	`if: github.ref == 'refs/heads/main'`
Run even after a failure	`if: always()`