REFERÊNCIA RÁPIDA DE GIT

Branches, merge, rebase, stash, remotos

Configuração

Configuração de Usuário

```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
git config --global core.editor vim
git config --list     # show all settings
```

Init & Clone

```
git init              # new repo in current dir
git clone <url>       # copy remote repo
git clone <url> dir   # clone into dir/
```

Básico

Status & Staging

```
git status            # working tree status
git add file.txt      # stage a file
git add .             # stage all changes
git add -p            # stage interactively
```

Commit

```
git commit -m "msg"   # commit staged changes
git commit -am "msg"  # stage tracked + commit
git commit --amend    # edit last commit
```

Diff

```
git diff              # unstaged changes
git diff --staged     # staged changes
git diff HEAD~1       # vs previous commit
git diff branchA branchB
```

Branches

Gerenciamento de Branches

```
git branch            # list local branches
git branch -a         # list all (incl. remote)
git branch feat       # create branch
git branch -d feat    # delete (safe)
git branch -D feat    # delete (force)
```

Trocar de Branch

```
git checkout feat     # switch to branch
git checkout -b feat  # create + switch
git switch feat       # switch (modern)
git switch -c feat    # create + switch (modern)
```

Merge

```
git merge feat        # merge feat into current
git merge --no-ff feat # always create merge commit
git merge --abort     # cancel conflicted merge
```

Remoto

Gerenciar Remotos

```
git remote -v               # list remotes
git remote add origin <url>  # add remote
git remote remove origin     # remove remote
```

Fetch, Pull, Push

```
git fetch origin      # download updates
git pull              # fetch + merge
git pull --rebase     # fetch + rebase
git push origin main  # push to remote
git push -u origin feat # push + set upstream
```

Log

Ver Histórico

```
git log               # full commit log
git log --oneline     # compact one-line format
git log --oneline -10 # last 10 commits
git log --graph --oneline --all # branch graph
git log -p            # show patches (diffs)
git log --stat        # show file change stats
```

Filtrar Log

```
git log --author="Alice"
git log --since="2024-01-01"
git log -- path/file  # commits touching file
git log -S "keyword"  # search content changes
```

Stash

```
git stash             # save working changes
git stash push -m "wip" # save with message
git stash list        # list all stashes
git stash pop         # apply + remove top stash
git stash apply       # apply but keep stash
git stash drop        # remove top stash
git stash clear       # remove all stashes
```

Desfazer Mudanças

Unstage & Descartar

```
git restore file.txt           # discard working changes
git restore --staged file.txt  # unstage file
git checkout -- file.txt       # discard (legacy)
```

Reset

`git reset --soft HEAD~1`	Desfaz commit, mantém staged
`git reset --mixed HEAD~1`	Desfaz commit, mantém working (padrão)
`git reset --hard HEAD~1`	Desfaz commit, descarta todas as mudanças

Revert

```
git revert <commit>   # new commit undoing changes
git revert HEAD       # undo last commit (safe)
```

Tags

```
git tag v1.0                  # lightweight tag
git tag -a v1.0 -m "Release" # annotated tag
git tag                       # list tags
git push origin v1.0          # push single tag
git push origin --tags        # push all tags
git tag -d v1.0               # delete local tag
```

.gitignore

Exemplos de Padrões

```
# .gitignore
*.log           # all .log files
build/          # build directory
!important.log  # exception (do track)
/TODO           # only root TODO
doc/**/*.pdf    # pdfs in doc/ subtree
```

Padrões Comuns

`*.ext`	Todos os arquivos com essa extensão
`dir/`	Diretório inteiro
`!pattern`	Negação (reincluir)
`**/name`	Corresponde em qualquer subdiretório
`?`	Curinga de um único caractere