Git cheat sheet

Configuration

├ 📌 **Set user name:** *git config --global user.name "Your Name"*

├ 📌 **Set email:** *git config --global user.email "youremail@example.com"*

├ 📌 **Check settings:** *git config --list*

├ 📌 **Set default editor:** *git config --global core.editor nano*

├ 📌 **Set merge tool:** *git config --global merge.tool vimdiff*

└ 📌 **Set diff tool:** *git config --global diff.tool meld*

Repository Setup

├ 📌 **Create new repository:** *git init*

├ 📌 **Clone a repository:** *git clone <repo-url>*

├ 📌 **Add remote repository:** *git remote add <name> <repo-url>*

└ 📌 **List remote repositories:** *git remote -v*

Basic Workflow

├ 📌 **Check status:** *git status*

├ 📌 **Add changes to staging area:** *git add <file>*

├ 📌 **Commit changes:** *git commit -m "Commit message"*

├ 📌 **Push changes:** *git push* **Pull changes:** *git pull*

├ 📌 **Fetch changes:** *git fetch*

Git Ultimate Guide

@coding_knowladge

Branching

├ 📌 **Create new branch:** *git branch <branch-name>*

├ 📌 **Switch to branch:** *git checkout <branch-name>*

├ 📌 **Create and switch to branch:** *git checkout -b <branch-name>*

├ 📌 **List branches:** *git branch*

├ 📌 **Merge branch into current branch:** *git merge <branch-name>*

├ 📌 **Delete branch:** *git branch -d <branch-name>*

└ 📌 **Force delete branch:** *git branch -D <branch-name>*

Stashing

├ 📌 **Save changes to stash:** *git stash save "Message"*

├ 📌 **List stashes:** *git stash list*

├ 📌 **Apply latest stash:** *git stash apply*

├ 📌 **Apply specific stash:** *git stash apply stash@{2}*

├ 📌 **Delete latest stash:** *git stash drop*

└ 📌 **Delete specific stash:** *git stash drop stash@{2}*

Tagging

├ 📌 **Create annotated tag:** *git tag -a <tag-name> -m "Message"*

├ 📌 **Create lightweight tag:** *git tag <tag-name>*

├ 📌 **List tags:** *git tag*

├ 📌 **Push tag to remote:** *git push --tags*

└ 📌 **Delete tag:** *git tag -d <tag-name>*

Git Ultimate Guide

@coding_knowladge

Undoing Changes

├ 📌 **Discard changes in working directory:** *git checkout -- <file>*

├ 📌 **Unstage file:** *git reset HEAD <file>*

├ 📌 **Undo commit:** *git reset HEAD~*

├ 📌 **Discard all changes since last commit:** *git reset --hard HEAD*

└ 📌 **Revert commit:** *git revert <commit-hash>*

Miscellaneous

├ 📌 **Show commit history:** *git log*

├ 📌 **Show differences between files:** *git diff <file>*

├ 📌 **Show differences between commits:** *git diff <commit1> <commit2>*

├ 📌 **Create a new commit from selected changes:** *git cherry-pick <commit-hash>*

├ 📌 **Configure global ignore file:** *git config --global core.excludesfile ~/.gitignore_global*

└ 📌 **Show file history:** *git log -- <file>*

Git Flow

├ 📌 **Initialize Git Flow:** *git flow init*

├ 📌 **Start new feature:** *git flow feature start <feature-name>*

├ 📌 **Finish feature:** *git flow feature finish <feature-name>*

├ 📌 **Start new release:** *git flow release start <version>*

├ 📌 **Finish release:** *git flow release finish <version>*

├ 📌 **Start new hotfix:** *git flow hotfix start <version>*

├ 📌 **Finish hotfix:** *git flow hotfix finish <version>*

└ 📌 **Publish branch:** *git flow publish <branch>*

Git Ultimate Guide

Git Aliases

├ 📌 **Create alias:** `git config --global alias.<alias-name> "<command>"`

├ 📌 **List aliases:** `git config --global --get-regexp alias`

└ 📌 **Delete alias:** `git config --global --unset alias.<alias-name>`

Git Hooks

├ 📌 **Client-side hooks:** `.git/hooks/`

├ 📌 **Server-side hooks:** `hooks/`

└ 📌 **Sample hook:** `pre-commit`

Advanced Workflow

├ 📌 **Rebase:** `git rebase <branch>`

├ 📌 **Squash commits:** `git rebase -i HEAD~<number-of-commits>`

├ 📌 **Interactive rebase:** `git rebase -i <commit-hash>`

├ 📌 **Cherry-pick commit range:** `git cherry-pick <start-commit>..<end-commit>`

├ 📌 **Merge with merge commit:** `git merge --no-ff <branch>`

├ 📌 **Merge with fast-forward:** `git merge --ff-only <branch>`

├ 📌 **Resolve merge conflicts:** `git mergetool`

├ 📌 **Bisect:** `git bisect <start> <end>`

└ 📌 **Reset branch to previous commit:** `git reset <commit>`

Harry
coding_knowladge

Git Ultimate Guide

@coding_knowladge

Git Internals

├ 📌 **Object model:** *blobs, trees, commits, tags*

├ 📌 **Git directory structure:** *objects, refs, HEAD*

├ 📌 **Object hash:** *git hash-object <file>*

├ 📌 **Create Git object:** *echo <content> | git hash-object -w --stdin*

├ 📌 **Git index:** *staging area and cache*

├ 📌 **Packfiles:** *compressed object storage*

├ 📌 **Git object storage optimization:** *git gc*

├ 📌 **Git hooks internals:** *pre-commit, post-commit, pre-push*

└ 📌 **Git objects with multiple parents:** *merge commits*

Git Graphical Clients

├ 📌 GitHub Desktop

├ 📌 SourceTree

├ 📌 GitKraken

├ 📌 Git Extensions

└ 📌 TortoiseGit

Harry
coding_knowladge

Git Ultimate Guide

Tips & Tricks

├ 📌 **Interactive staging:** *git add -p*

├ 📌 **Amend commit message:** *git commit --amend*

├ 📌 **Show commit changes:** *git show <commit-hash>*

├ 📌 **Show changes from branch to another:** *git diff <branch1>..<branch2>*

├ 📌 **Show local branches tracking remote branches:** *git branch -vv*

├ 📌 **Show repository statistics:** *git diff --stat*

└ 📌 **Show who last modified each line of a file:** *git blame <file>*

Best Practices

├ 📌 Keep commits small and focused

├ 📌 Write meaningful commit messages

├ 📌 Use feature branches for development

├ 📌 Keep the master branch stable

├ 📌 Pull before pushing

├ 📌 Use Git Flow for complex workflows

├ 📌 Use Git Hooks for automation

└ 📌 Use Git Aliases for frequently used commands

Harry

coding_knowladge

Resources

├ Official Git website: https://git-scm.com/

├ Git documentation: https://git-scm.com/doc

├ Pro Git book: https://git-scm.com/book/en/v2

├ GitHub Learning Lab: https://lab.github.com/

├ Atlassian Git tutorial: https://www.atlassian.com/git/tutorials

├ Git Immersion: https://gitimmersion.com/

├ Git Cheat Sheet by GitHub: https://education.github.com/git-cheat-sheet-education.pdf

└ Git Cheat Sheet by Tower: https://www.git-tower.com/blog/git-cheat-sheet/