THE ULTIMATE GIT CHEAT SHEET

Set Up and Create a Repository

Set your commit identity, start a repository, or clone an existing one.

```
# confirm Git is installed
# short version: git -v
$ git --version
## => output
### git version 2.51.0

# identify your commits
$ git config --global user.name "Your Name"
$ git config --global user.email you@example.com
$ git config --global init.defaultBranch main

# short version: git config -l --show-origin
$ git config --list --show-origin
## => output
### file:/home/u/.gitconfig user.name=Your Name
### file:/home/u/.gitconfig user.email=you@example.com
### file:/home/u/.gitconfig init.defaultbranch=main

# start a project
$ git init
## => output
### Initialized empty Git repository in /project/.git/

# copy a project
$ git clone <url>

$ git status # inspect before changing

# show help for the commit command
$ git help commit
```

Track and Commit Changes

Move changes from your working directory into the staging area, review them, and save an understandable snapshot with a commit.

```
# working directory -> staging area -> commit
# unstaged changes, only exist in working directory
$ git diff

$ git add <file> # stage one file

# interactively choose changes for the next commit
# short version: git add -p
$ git add --patch

$ git diff --staged # review next commit

# short version: git commit -m "Message"
$ git commit --message "Message"
## => output
### [main d4e5f6a] Message

# inspect a path ignored by .gitignore
# short version: git check-ignore -v .env
$ git check-ignore --verbose .env
## => output
### .gitignore:1:.env  .env
```

Inspect History and Compare

Git history is a chain of commits. Use log, show, blame, and diff to understand when a change happened and how snapshots differ.

```
$ git log --oneline

$ git log --oneline --graph --decorate --all
## => output
### * d4e5f6a (HEAD -> main) Message
### * a1b2c3d Initial commit

$ git show <commit> # commit and patch

$ git blame README.md # last line authors

$ git diff HEAD # all local changes

$ git diff main..some-feature-branch

$ git log main..some-feature-branch --oneline
## => output
### b7c8d9e Feature branch message
```

Create Branches and Merge

Branches let you develop an idea without changing main. Switch to a branch, commit there, and merge the completed work back.

```
$ git branch # list branches
## => output
### * main
###   some-feature-branch

# short version: git switch -c <branch>
$ git switch --create <branch> # create + switch
## => output
### Switched to a new branch 'some-feature-branch'

# switch without creation
$ git switch <branch>

$ git switch - # previous branch

# merge <branch> into current branch
$ git merge <branch>
## => output
### Updating d4e5f6a..b7c8d9e
### Fast-forward

$ git merge --abort # cancel conflict

# short version: git branch -d <branch>
$ git branch --delete <branch> # delete if merged
## => output
### Deleted branch some-feature-branch.
```

Resolve a Merge Conflict

When Git cannot combine two edits automatically, inspect the conflicted file, remove its markers, stage the resolution, and commit it.

```
# 1. find files marked "both modified"
$ git status
## => output
### both modified: <file>

# 2. edit the file and remove <<<, === and >>>
# 3. stage the resolution, verify and finish
$ git add <file>

# - is a removed line
# + is an added line
$ git diff --staged
## => output
### -old line
### +new line

# 4. finish the merge with a commit
$ git commit
# or cancel the merge and restore the pre-merge state
$ git merge --abort
```

Fetch, Pull, and Push

A remote is a shared repository. Fetch downloads its history, pull updates your branch, and push publishes your local commits.

```
# short version: git remote -v
$ git remote --verbose # inspect remotes
## => output
### origin https://github.com/user/project.git (fetch)
### origin https://github.com/user/project.git (push)

$ git remote add origin <url>

$ git remote show origin

$ git fetch origin # download only

$ git pull # fetch and integrate

$ git pull --ff-only # update, no merge

# short version: git pull -r
$ git pull --rebase # fetch, then reapply local commits

# short version: git push -u origin <branch>
$ git push --set-upstream origin <branch>

$ git push # push new commits to the same branch
```

Blog: https://devopscycle.com/blog/the-ultimate-git-cheat-sheet/
GitHub: https://github.com/aichbauer/the-ultimate-git-cheat-sheet/
Consulting: https://devopscycle.com/