Git Commands Cheat Sheet

Contents

Configuration

Set up global config

- Set your name:
  `git config --global user.name "Your Name"`
- Set your email:
  `git config --global user.email "you@example.com"`
- Check current settings:
  `git config --list`

Repository Management

Initialize and Clone

- Initialize a new Git repository:
  `git init`
- Clone an existing repository:
  `git clone <repository_url>`

Staging and Committing

Adding and Committing Changes

- Add a specific file to the staging area:
  `git add file.txt`
- Add all changes to the staging area:
  `git add .`
- Commit changes with a message:
  `git commit -m "Your commit message"`
- Commit all changes directly (skip staging):
  `git commit -a -m "Your commit message"`

Branch Management

Working with Branches

- List all branches:
  `git branch`
- Create a new branch:
  `git branch new-branch`
- Switch to a branch:
  `git checkout branch-name`
- Create and switch to a branch:
  `git checkout -b new-branch`
- Rename the current branch:
  `git branch -m new-name`
- Delete a branch:
  `git branch -d branch-name`

Merging and Rebasing

Combining Changes

- Merge a branch into the current branch:
  `git merge branch-name`
- Rebase the current branch onto another branch:
  `git rebase branch-name`

Remote Repositories

Connecting to a Remote

- Add a remote repository:
  `git remote add origin <repository_url>`
- List remote repositories:
  `git remote -v`
- Remove a remote repository:
  `git remote remove origin`

Pushing and Pulling

- Push changes to a remote repository:
  `git push origin branch-name`
- Pull changes from a remote repository:
  `git pull origin branch-name`
- Push all branches to the remote:
  `git push --all origin`
- Push tags to the remote:
  `git push --tags`

Status and Logs

Checking Repository State

- Check the status of the working directory:
  `git status`

- View commit history:
  `git log`
- View a simplified commit history:
  `git log --oneline`
- View changes in files:
  `git diff`

Stash Management

Save and Restore Work in Progress

- Stash changes:
  `git stash`
- List stashed changes:
  `git stash list`
- Apply the most recent stash:
  `git stash apply`
- Apply and remove the most recent stash:
  `git stash pop`
- Clear all stashes:
  `git stash clear`

Tagging

Annotating Versions

- Create a lightweight tag:
  `git tag tag-name`
- Create an annotated tag:
  `git tag -a tag-name -m "Tag message"`
- Push tags to a remote:
  `git push origin tag-name`
- List all tags:
  `git tag`
- Delete a tag:
  `git tag -d tag-name`

Undoing Changes

Revert, Reset, and Clean

- Undo the last commit (keep changes):
  `git reset HEAD~1`
- Undo the last commit (discard changes):
  `git reset --hard HEAD~1`
- Remove untracked files:
  `git clean -f`
- Revert a commit:
  `git revert commit-hash`

Collaboration

Resolving Conflicts

- Start a merge with conflicts:
  `git merge branch-name`
- Resolve conflicts in files, then mark resolved:
  `git add file.txt`
- Continue after resolving conflicts:
  `git merge --continue`

Advanced Commands

Cherry-Pick and Bisect

- Apply a specific commit to the current branch:
  `git cherry-pick commit-hash`
- Start a bisect to find a buggy commit:
  `git bisect start`
- Mark the current commit as good:
  `git bisect good`
- Mark the current commit as bad:
  `git bisect bad`

Squash Commits

- Interactively rebase to squash commits:
  `git rebase -i HEAD~n`

Aliases

Create Custom Shortcuts

- Create a custom alias:
  `git config --global alias.co checkout`
- Use the alias:
  `git co branch-name`

Tips and Tricks

Useful Options

- Clone only the latest commit:
  `git clone --depth 1 <repository_url>`
- List files changed in a commit:
  `git show --name-only commit-hash`
- Show changes for a specific file:
  `git log -p file.txt`