Git & GitHub – Complete Guide

Repository Initialization

git init – creates a new local repository
git clone <url> – downloads repository from GitHub

Working with Files

git status – shows status of changes
git add <file> – adds file to staging area
git add . – adds all modified files
git commit -m "message" – saves changes to history

Pushing and Pulling

git push – sends changes to remote repository
git pull – fetches and merges changes from GitHub
git fetch – only fetches changes without merging

On GitHub you can accept pull requests through the web interface.

Branching

git branch – shows list of branches
git branch <name> – creates new branch
git checkout <name> – switches to branch
git switch <name> – alternative to checkout
git checkout -b <name> – creates and switches to new branch

In GitHub interface you can create new branch via Branches → New branch.

Merge and Rebase

git merge <branch> – merges branch with current one
git rebase <branch> – rewrites history on top of branch

Resetting Commits

git reset --soft HEAD~1 – undoes commit, keeps changes in staging
git reset --mixed HEAD~1 – undoes commit and staging (default option)

git reset --hard HEAD~1 – undoes commit and removes all changes
git revert <commit> – creates commit that reverses changes without removing history

History and Logs

git log – shows complete commit history
git log --oneline – condensed version of history
git diff – shows differences between versions

Cleaning and Removing

git clean -fd – removes untracked files and directories
git rm <file> – removes file from repository

Remote Repositories

git remote -v – shows configured remote repositories
git remote add origin <url> – adds new remote repository
git push -u origin <branch> – pushes changes and sets up branch tracking

Advanced History Operations

git rebase -i HEAD~N – interactive editing of last N commits
git push --force – forces overwrite of remote repository history
git push --force-with-lease – safer history overwrite

GitHub Interface Features

Through GitHub web interface you can perform the following operations:

- Create new repositories
- Create and manage branches
- Merge pull requests
- Review commit history
- Edit files online
- Manage collaborators

Staging and Detailed Comparisons

git diff – differences between working directory and staging area
git diff --cached – differences between staging area and last commit
git diff <branch1>..<branch2> – differences between branches

git restore <file> – restores file to version from last commit
git restore --staged <file> – removes file from staging area

Interactive File Operations

git add -p – allows selecting file fragments for commit
git stash – temporarily stores changes
git stash pop – restores last stashed changes

Reflog - Recovering Lost Commits

git reflog – shows history of all Git operations
git checkout <hash> – return to specific commit from reflog

Cherry-pick and Bisect

git cherry-pick <hash> – moves specific commit to current branch
git bisect start – begins search for buggy commit
git bisect good/bad <hash> – marks commit as working or buggy

GPG Signatures

git commit -S -m "msg" – creates signed commit with GPG key

GitHub marks signed commits as "Verified".

Version Tagging

git tag v1.0 – creates tag locally
git push origin v1.0 – pushes tag to GitHub
git tag -d v1.0 – deletes tag locally
git push origin :refs/tags/v1.0 – deletes tag from remote repository

Pull Requests and Forks

On GitHub you can fork repository using "Fork" button and create pull request via "New pull request".

gh pr create – creates pull request from command line (GitHub CLI)

SSH and Authorization

ssh-keygen -t ed25519 – generates SSH key

Public key should be added in GitHub → Settings → SSH & GPG keys.

Submodules

git submodule add <url> – adds another repository as submodule
git submodule update --init --recursive – fetches and initializes all submodules

Configuration

git config --global user.name "First Last" – global settings
git config --local user.email "email@example.com" – settings for specific repo
git config --list – displays all settings

Synchronization with Main Repository

git fetch upstream – fetches data from main repository
git rebase upstream/main – rewrites local branch on latest version

Pull Request Workflow

bash

```
git checkout -b new-feature
# make changes
git commit -m "added new feature"
git push origin new-feature
# then on GitHub: Compare & pull request
```

Best Practices

- Always run `git pull` before `git push`
- Avoid `--force` in team projects without consultation
- Create separate branches for new features
- Make frequent commits with descriptive messages
- Use meaningful names for branches and commits

Useful Aliases

Add to `.gitconfig` file:

ini

```ini
[alias]
  st = status
  ci = commit
  co = checkout
  br = branch
  lg = log --oneline --graph --decorate
```