git config

`git config --global user.name <"name">`	postavljanje korisničkog imena
`git config --global user.email <"email">`	postavljanje korisničkog emaila
`git config --global core.editor "vim --nofork"`	postavljanje Vim-a kao editora
`git config --global core.editor "atom --wait"`	postavljanje Atom-a kao editora
`git config --global core.editor "code --wait"`	postavljanje VS Code-a kao editora

git init

`git init`	inicijalizacija trenutnog direktorija kao repozitorija

git clone

`git clone <URL>`	kloniranje repozitorija

git remote

`git remote [-v]`	[detaljni] prikaz *remote*-ova
`git remote add upstream <URL>`	dodavanje *upstream remote*-a (za *fetch*-anje *update*-ova na originalnom repozitoriju)
`git remote add origin <URL>`	dodavanje *upstream remote*-a (za *push*-anje *update*-ova na *forked* repozitorij)
`git remote rename <old> <"new">`	preimenovanje *remote*-a
`git remote remove <name>`	uklanjanje *remote*-a

git fetch

`git fetch <remote>`	*update*-a stanje svih *branch*-eva *remote* repozitorija
`git fetch <remote> <remote-branch>`	*update*-a stanje pojedinačnog *branch*-a *remote* repozitorija

git pull

- `git pull origin <remote-branch>` = `git fetch origin <remote-branch>` + `git merge <remote>/<branch>`

`git pull [-X ours|theirs] <remote> <remote-branch>`	*update*-a stanje pojedinačnog *branch*-a remote repozitorija i sukladno *update*-a lokalno stanje [uz prihvaćanje isključivo *naših* ili isključivo *njihovih* promjena]

git push

`git push [-f] <remote> <local-branch>`	[nasilno] *push*-anje lokalnog *branch*-a na *remote* istog naziva
`git push --tags <remote> <local-branch>`	*push*-a tagove na remote repozitorij (*tag*-ovi se neće *push*-ati automatski)

git status

`git status`	status repozitorija (*branch*, *untracked*, *unstaged*, *staged*...)

git log

`git log`	puni ispis svih *commit*-a na repozitoriju
`git log --oneline`	jednolinijski ispis svih *commit*-a na repozitoriju

git add

`git add <file>`	dodavanje datoteka u *staging area*

git commit

`git commit`	*commit staged* datoteka (naknadni unos *commit* poruke)
`git commit -m <message>`	*commit staged* datoteka uz poruku
`git commit -a -m <message>`	dodavanje svog sadržaja radnog direktorija u staging area i *commit* uz poruku
`git commit --amend`	*redo* posljednjeg *commit*-a ili uređivanje poruke posljednjeg *commit*-a

git branch

`git branch -l [-v]`	[detaljni] prikaz lokalnih *branch*-eva
`git branch -r [-v]`	[detaljni] prikaz remote *branch*-eva
`git branch -a [-v]`	[detaljni] prikaz svih *branch*-eva
`git branch <branch-name>`	kreiranje novog *branch*-a
`git branch -m <branch-name>`	preimenovanje aktivnog *branch*-a (koristiti `-M` za nasilno)
`git branch -d <branch-name>`	brisanje *branch*-a (koristiti `-D` za nasilno)

git switch

`git switch <branch-name>`	prebacivanje s trenutnog na drugi *branch* (ako se koristi ime *remote branch*-a, kreira lokalnu kopiju)
`git switch -`	prebacivanje u prethodno posjećeni *branch*
`git switch -c <branch-name>`	kreiranje novog *branch*-a s imenom name s trenutnog *HEAD*-a i prebacivanje u njega

git merge

- *merge*-ani *branch* se ne briše

`git merge <branch-name>`	*merge*-anje *branch*-a u aktivni *branch*

git rebase

- rebase funkcijski može služiti kao zamjena za *merge*
  - reorganizira/kopira *commit* s aktualnog *branch*-a na vrh master *branch*-a

`git rebase main`	reorganizacija/kopiranje *commit*-a s aktualnog *branch*-a na vrh main *branch*-a
`git rebase -i <commit-hash>`	ulazak u interaktivno uređivanje prethodnih *commit*-a, počevši od *commit*-a prije *commit-hash*-a

git diff

`git diff <file>`	prikazivanje *unstaged* promjena datoteke
`git diff --staged <file>`	prikazivanje *staged* promjena datoteke
`git diff HEAD <file>`	prikazivanje svih promjena datoteke od posljednjeg *commit*-a
`git diff <old-commit-hash> <new-commit-hash>`	prikazivanje promjena između dva *commit*-a
`git diff <branch-name-1> <branch-name-2>`	prikazivanje promjena između "vrhova" *branch*-eva
`git diff <branch-name>@{0} <branch-name>@{1}`	prikazuje promjene između stanja na *branch*-u (vidi `git reflog show`)

git clean

`git clean -fdx [-n]`	obriši sve *untracked* datoteke i direktorije, rekurzivno (opcija `-n` samo prikazuje što će se obrisati)

git stash

- *stash* je globalan, nije vezan za *branch*
  - moguće je aplicirati *stashed* promjene s jednog na drugi *branch*

`git stash save`	spremanje *unstaged* i *staged* promjena u *stash*
`git stash list`	listanje *stashed* promjena
`git stash pop [stash@<number>]`	apliciranje *stashed* promjena na radni direktorij (cut iz *stash*-a)
`git stash apply [stash@<number>]`	apliciranje *stashed* promjena na radni direktorij (copy iz *stash*-a)
`git stash drop [stash@<number>]`	brisanje *stashed* promjene
`git stash clear`	brisanje svih *stashed* promjena iz liste

git checkout

`git checkout <branch-name>`	prebacivanje s trenutnog *HEAD*-a na drugi *branch* (funkcionira i za *remote branch*-eve)
`git checkout -b <branch-name>`	kreiranje novog *branch*-a s imenom name s trenutnog *HEAD*-a i prebacivanje u njega
`git checkout --track <remote>/<branch>`	kreira lokalnu kopiju *remote branch*-a istog naziva (vidi `git switch`)
`git checkout <commit-hash>`	prebacivanje *HEAD*-a na neki drugi *commit*
`git checkout <tag>`	prebacivanje *HEAD*-a na neki drugi *commit* označen *tag*-om
`git checkout HEAD <file>`	poništavanje svih promjena do posljednjeg *commit*-a za datoteku
`git checkout HEAD@{<number>}`	premještanje u poziciju *HEAD*-a prije *number* koraka (vidi `git reflog show`)
`git checkout <branch-name>@{<number>}`	premještanje u stanje *branch*-a prije *number* koraka (vidi `git reflog show`)

git tag

- *tag* je dodatan pointer na *commit*
  - *tag*-ovi su jedinstveni
- uglavnom služe za *semantic versioning*

`git tag -l ["<regex-search>"]`	ispisuje sve *tag*-ove na aktualnom *branch*-u [koji zadovoljavaju *regex* pretragu]
`git tag [-a] <"tag-name">`	tagira aktualni *HEAD*/*commit* s [*annotated*] *tag*-om
`git tag [-a] <"tag-name"> <commit-hash>`	tagira *commit* s [*annotated*] *tag*-om
`git tag -f <"tag-name"> <commit-hash>`	"nasilno" tagira *commit* s već iskorištenim *tag*-om (tag sa starog *commit*-a nestaje)
`git tag -d <tag-name>`	briše *tag*

git restore

- nikad ne uklanja *commit log*

`git restore --staged <file>`	prebacivanje *staged* promjena u *unstaged*
`git restore <file>`	poništavanje *unstaged* promjena do posljednjeg *commit*-a
`git restore --source=<commit-hash> <file>`	poništavanje promjena po uzoru na *commit* (*log* ostaje isti, promjene odlaze u *unstaged*)

git reset

- uvijek uklanja *commit log*
- *unstaged* i *staged* promjene se mogu prebaciti na drugi branch

`git reset --soft <commit-hash>`	poništavanje promjena po uzoru na *commit* (*log* se mijenja, promjene odlaze u *staged*)
`git reset [--mixed] <commit-hash>`	poništavanje promjena po uzoru na *commit* (*log* se mijenja, promjene odlaze u *unstaged*)
`git reset --hard <commit-hash>`	poništavanje promjena po uzoru na *commit* (*log* se mijenja, promjene se *commit*-aju)

git revert

- nikad ne uklanja *commit log*

`git revert <commit-hash>`	poništavanje promjena od *commit*-a i dalje (*log* ostaje isti, promjene odlaze u *unstaged-conflicted*)

git reflog

- *reflog*-ovi su lokalni i imaju "rok trajanja"
- služe za poništavanje `git reset` i sl.

`git reflog show HEAD`	prikazuje povijest "kretanja" *HEAD*-a
`git reflog show <branch-name>`	prikazuje povijest stanja na *branch*-u
`git reflog show HEAD@2.day.ago`	prikazuje povijest "kretanja" *HEAD*-a do prije 2 dana
`git reflog show <branch-name>@2.day.ago`	prikazuje povijest stanja na *branch*-u do prije 2 dana