

## git config

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

## git init

<code>git init</code>	inicijalizacija trenutnog direktorija kao repozitorija
-----------------------	--

## git clone

<code>git clone &lt;URL&gt;</code>	kloniranje repozitorija
------------------------------------	-------------------------

## git remote

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

## git fetch

<code>git fetch &lt;remote&gt;</code>	<i>update</i> -a stanje svih <i>branch</i> -eva <i>remote</i> repozitorija
<code>git fetch &lt;remote&gt; &lt;remote-branch&gt;</code>	<i>update</i> -a stanje pojedinačnog <i>branch</i> -a <i>remote</i> repozitorija

## git pull

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

<code>git pull &lt;remote&gt; &lt;remote-branch&gt;</code>	<i>update</i> -a stanje pojedinačnog <i>branch</i> -a remote repozitorija i sukladno <i>update</i> -a lokalno stanje
--	--

## git push

<code>git push &lt;remote&gt; &lt;local-branch&gt;</code>	<i>push</i> -anje lokalnog <i>branch</i> -a na <i>remote</i> istog naziva
<code>git push --tags &lt;remote&gt; &lt;local-branch&gt;</code>	<i>push</i> -a tagove na remote repozitorij ( <i>tag</i> -ovi se neće <i>push</i> -ati automatski)

## git status

<code>git status</code>	status repozitorija ( <i>branch</i> , <i>untracked</i> , <i>unstaged</i> , <i>staged</i> ...)
-------------------------	---

## git log

<code>git log</code>	puni ispis svih <i>commit</i> -a na repozitoriju
<code>git log --oneline</code>	jednolinijski ispis svih <i>commit</i> -a na repozitoriju

## git add

<code>git add &lt;file&gt;</code>	dodavanje datoteka u <i>staging area</i>
-----------------------------------	--

## git commit

<code>git commit</code>	<i>commit staged</i> datoteka (naknadni unos <i>commit</i> poruke)
<code>git commit -m &lt;message&gt;</code>	<i>commit staged</i> datoteka uz poruku
<code>git commit -a -m &lt;message&gt;</code>	dodavanje svog sadržaja radnog direktorija u staging area i <i>commit</i> uz poruku
<code>git commit --amend</code>	<i>redo</i> posljednjeg <i>commit</i> -a ili uređivanje poruke posljednjeg <i>commit</i> -a

## git branch

<code>git branch -l [-v]</code>	[detaljni] prikaz lokalnih <i>branch</i> -eva
<code>git branch -r [-v]</code>	[detaljni] prikaz remote <i>branch</i> -eva
<code>git branch -a [-v]</code>	[detaljni] prikaz svih <i>branch</i> -eva
<code>git branch &lt;branch-name&gt;</code>	kreiranje novog <i>branch</i> -a
<code>git branch -m &lt;branch-name&gt;</code>	preimenovanje aktivnog <i>branch</i> -a (koristiti <code>-M</code> za nasilno)
<code>git branch -d &lt;branch-name&gt;</code>	brisanje <i>branch</i> -a (koristiti <code>-D</code> za nasilno)

## git switch

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

## git merge

- *merge*-ani *branch* se ne briše

<code>git merge &lt;branch-name&gt;</code>	<i>merge</i> -anje <i>branch</i> -a u aktivni <i>branch</i>
--	---

## git rebase

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

<code>git rebase main</code>	reorganizacija/kopiranje <i>commit</i> -a s aktualnog <i>branch</i> -a na vrh main <i>branch</i> -a
<code>git rebase -i &lt;commit-hash&gt;</code>	ulazak u interaktivno uređivanje prethodnih <i>commit</i> -a, počevši od <i>commit-hash</i> -a

## git diff

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

## git stash

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

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

## git checkout

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

## git tag

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

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

## git restore

- nikad ne uklanja *commit log*

<code>git restore --staged &lt;file&gt;</code>	prebacivanje <i>staged</i> promjena u <i>unstaged</i>
<code>git restore &lt;file&gt;</code>	poništanje <i>unstaged</i> promjena do posljednjeg <i>commit</i> -a
<code>git restore --source=&lt;commit-hash&gt; &lt;file&gt;</code>	poništanje promjena po uzoru na <i>commit</i> ( <i>log</i> ostaje isti, promjene odlaze u <i>unstaged</i> )

## git reset

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

<code>git reset --soft &lt;commit-hash&gt;</code>	ponišćavanje promjena po uzoru na <i>commit</i> ( <i>log</i> se mijenja, promjene odlaze u <i>staged</i> )
<code>git reset [--mixed] &lt;commit-hash&gt;</code>	ponišćavanje promjena po uzoru na <i>commit</i> ( <i>log</i> se mijenja, promjene odlaze u <i>unstaged</i> )
<code>git reset --hard &lt;commit-hash&gt;</code>	ponišćavanje promjena po uzoru na <i>commit</i> ( <i>log</i> se mijenja, promjene se <i>commit</i> -aju)

## git revert

- nikad ne uklanja *commit log*

<code>git revert &lt;commit-hash&gt;</code>	ponišćavanje promjena od <i>commit</i> -a i dalje ( <i>log</i> ostaje isti, promjene odlaze u <i>unstaged-conflicted</i> )
---	--

## git reflog

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

<code>git reflog show HEAD</code>	prikazuje povijest "kretanja" <i>HEAD</i> -a   povijest radnji na <i>branch</i> -u
<code>git reflog show &lt;branch-name&gt;</code>	prikazuje povijest stanja na <i>branch</i> -u
<code>git reflog show HEAD@2.day.ago</code>	prikazuje povijest "kretanja" <i>HEAD</i> -a do prije 2 dana
<code>git reflog show &lt;branch-name&gt;@2.day.ago</code>	prikazuje povijest stanja na <i>branch</i> -u do prije 2 dana