Principi softverskog inženjerstva

GIT

Sistemi za kontrolu verzija

Sistemi za kontrolu verzija (*VCS - Version Control Systems*) su softverski alati koji omogućavaju praćenje promena u izvornom kodu (i drugim vrstama datoteka) tokom vremena. Nekoliko ključnih aspekata koje ovi sistemi omogućavaju su:

- **Praćenje promena** - Svaki put kada se napravi izmena u datoteci, VCS beleži tu promenu. Ovo uključuje dodavanje novih datoteka, brisanje postojećih datoteka i izmene u postojećim datotekama.
- **Verzionisanje** - VCS čuva potpunu istoriju promena. Programeri mogu videti kako je izgledao kod u prošlosti, kada su izmene napravljene i druge relevantne informacije u zavisnosti od konkretnog sistema.
- **Reverzibilnost** - VCS omogućava vraćanje projekta na prethodne verzije ukoliko je to potrebno.
- **Posmatranje razlika** - VCS omogućava uvid u razlike između različitih verzija koda, što je korisno za pronalaženje grešaka, analizu promena i održavanje kvaliteta koda.
- **Grananje i spajanje** - VCS omogućava kreiranje različitih grana u kojima se može raditi na različitim funkcionalnostima ili ispravkama. Ove grane se kasnije mogu spajati kako bi se integrisale promene.
- **Udaljeni repozitorijumi** – Većina sistema za kontrolu verzija podržava rad sa udaljenim repozitorijumima, koji omogućavaju laku saradnju između programera.

Neki od trenutno najpopularnijih sistema za kontrolu verzija su Mercurial, Subversion, CVS i Git.

Sistemi za kontrolu verzija se mogu klasifikovati u tri kategorije sa specifičnim karakteristikama:

- **Lokalni sistem za kontrolu verzija (*LVCS*)**

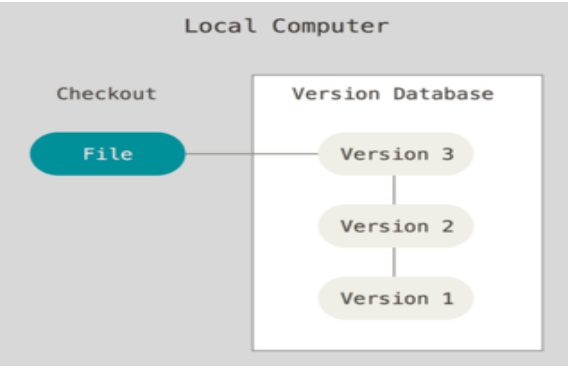
  Lokalni sistem za kontrolu verzija upravlja promenama nad datotekama u okviru lokalnog repozitorijuma na računaru korisnika. Nedostatak ovog sistema je to što su sve datoteke i baza verzija dostupne samo lokalno, pa korisnici koji nemaju pristup računaru na kojem se nalazi ne mogu da im pristupe. Takođe, ako se desi neki problem sa lokalnom bazom to može dovesti do gubitka ispraćenih promena i verzija.

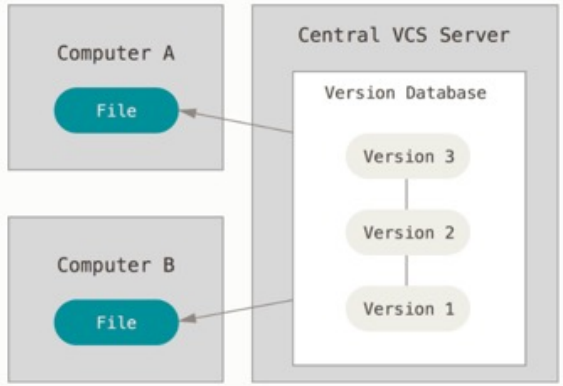
- **Centralizovani sistem za kontrolu verzija (*CVCS*)**

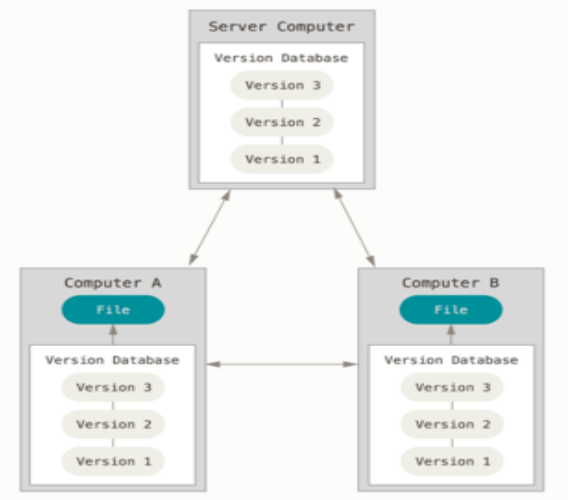
  Centralizovani sistem za kontrolu verzija skladišti i upravlja promenama nad datotekama u okviru centralnog repozitorijuma kojem može pristupiti više korisnika. U ovakvom sistemu korisnici prave različite promene nad datotekama u centralnom repozitorijumu koje sistem nadgleda i prati. Jedan od glavnih nedostataka jeste centralizovani server koji predstavlja centralnu tačku kvara.

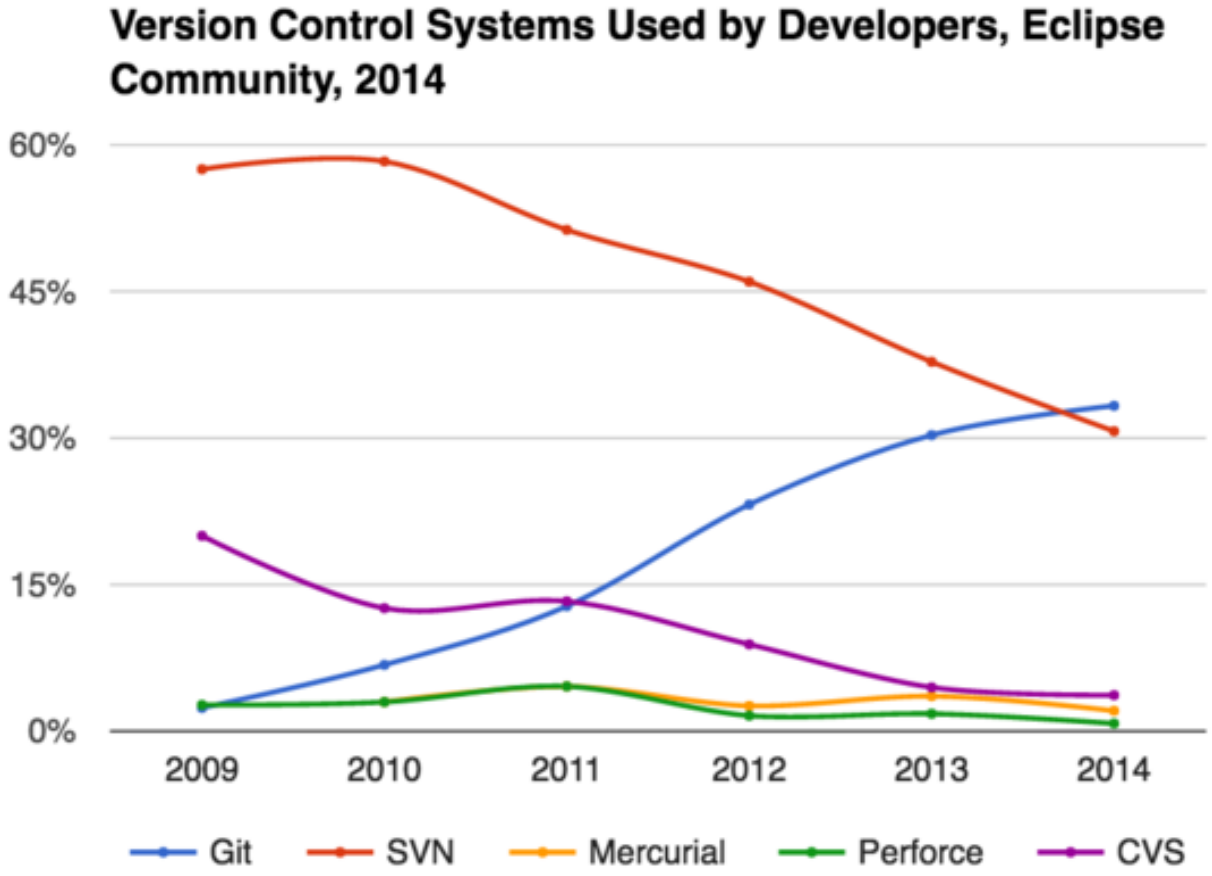
- **Distribuirani sistem za kontrolu verzija (*DVCS*)**

  Kod distribuiranih sistema za kontrolu verzija, svaki korisnik ima kompletnu kopiju baze projekta, uključujući celu istoriju promena pravljenih nad datotekama, što rešava problem centralne tačke kvara. Više korisnika može u isto vreme da pravi izmene nad granama lokalno, bez potrebe za povezivanjem na centralni server.







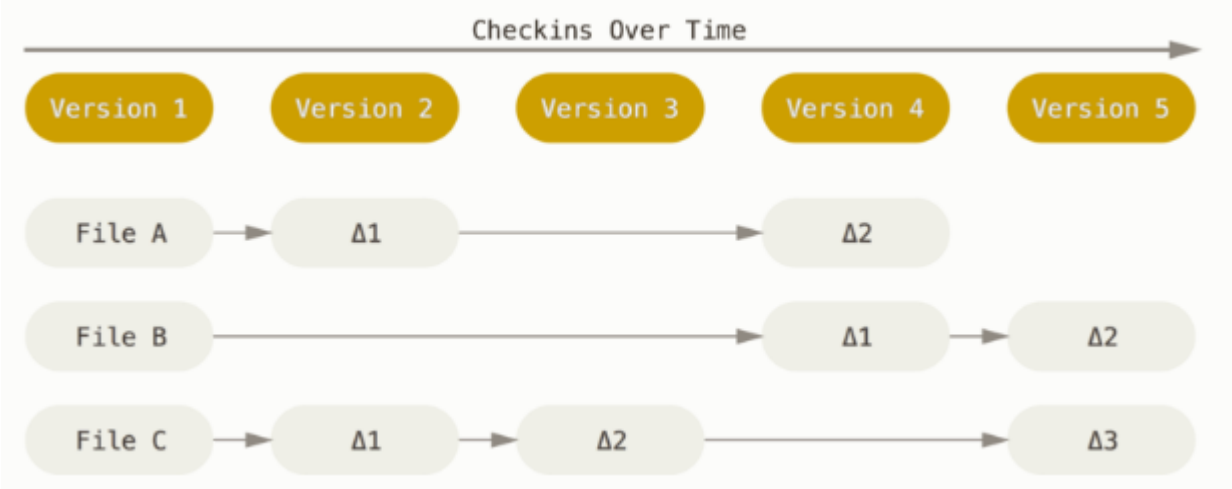


Šta je Git?

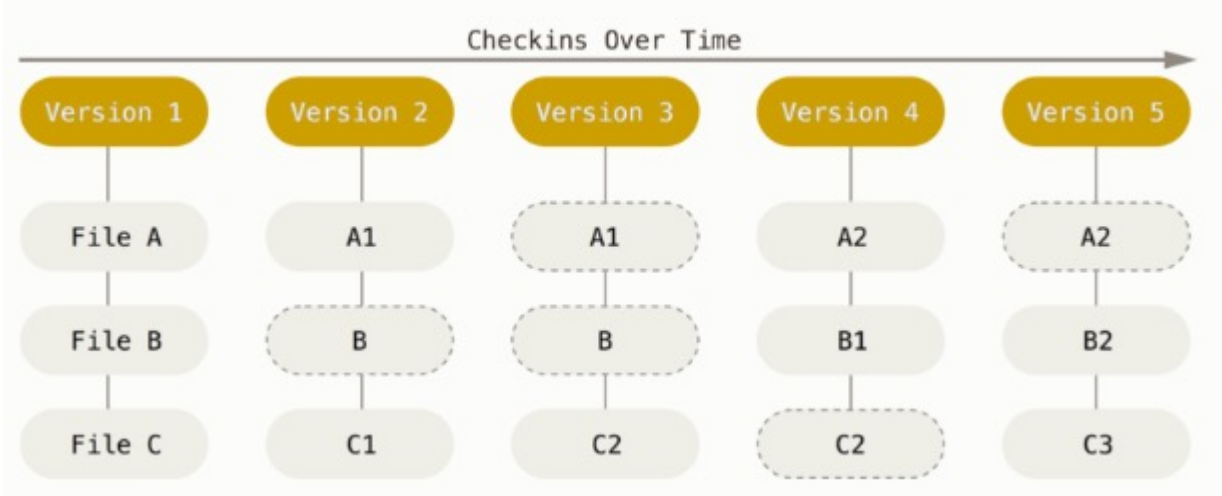
Git je distribuirani sistem za kontrolu verzija (može se koristiti i kao lokalni), kreiran od strane Linusa Torvaldsa 2005. godine. Ključne karakteristike *Git*-a koje ga razlikuju od drugih sistema za kontrolu verzija su:

- Distribuirana arhitektura
- Laka kolaboracija sa drugim ljudima
- Brzina i kod većih baza koda
- Bolje korišćenje prostora na disku
- Laka integracija sa različitim alatima

Razlika između *Git*-a i većine ostalih sistema za kontrolu verzija



Ostali sistemi - *delta-based version control*



Git – stream of snapshots

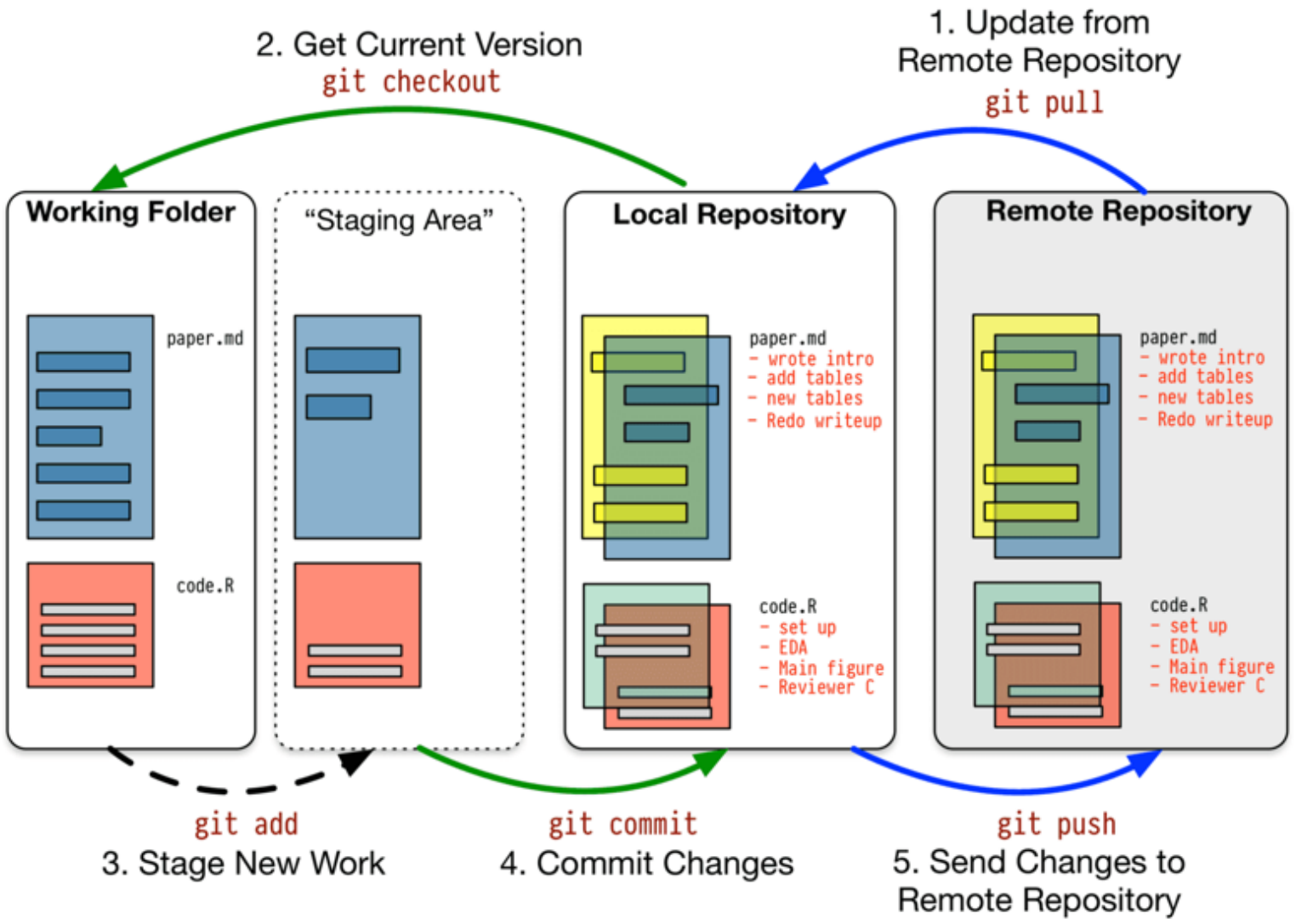
Još neke prednosti *Git*-a

Skoro sve operacije su lokalne – za većinu operacija su dovoljni samo podaci koji se nalaze na lokalnom repozitorujumu, čineći ove operacije dosta bržim u odnosu na operacije izvršavane kod centralizovanih sistema za kontrolu verzija.

Većina operacija može da se radi bez konekcije na udaljeni repozitorijum, jer su skoro sve operacije lokalne, pa čak i bez veze sa internetom mogu da se prave i prate promene učinjene nad projektom.

Integritet podataka – *checksum* se računa za sve promene i datoteke pre što promene bivaju sačuvane. Ovo znači da je nemoguće promeniti neku datoteku ili direktorijum, a da *Git* ne zna za to, odnosno ne mogu se izgubiti informacije u tranzitu kroz mrežu ili korupcijom datoteka, a da *Git* to ne detektuje.

Kako *Git* funkcioniše



Postoje tri stanja koja *Git* definiše u kojima datoteka može da se nalazi:

- *Modified* – datoteka je izmenjena ali ta izmena nije sačuvana u bazi lokalnog repozitorijuma.
- *Staged* - datoteka je izmenjena i markirana da uđe u sledeći *commit* snimak.
- *Commited* - datoteka je u trenutnoj verziji sačuvana u bazi lokalnog repozitorijuma.

- Radni direktorijum (eng. *Working Folder*) predstavlja jednu verziju na jednoj grani projekta. Komandom *checkout* se iz komprimovane baze u *Git* direktorijumu dovlače datoteke projekta i stavljaju u radni direktorijum, nakon čega se mogu koristiti i menjati.
- *Staging Area* (indeksna datoteka) predstavlja datoteku u okviru *Git* direktorijuma koji čuva koje datoteke će se "snimiti" u okviru sledećeg *commit*-a.
- Lokalni repozitorijum (*Git* direktorijum) je mesto u kojem *Git* čuva sve metapodatke i objektnu bazu podataka za tekući projekat. Predstavlja najbitniji deo za funkcionisanje *Git*-a, i upravo ovaj direktorijum se kopira sa udaljenog repozitorijuma komandom *clone*.

Korišćenje *Git*-a se obično sastoji iz tri koraka:

1. Izmena datoteka u okviru radnog direktorijuma.
2. Biranje datoteka koje ulaze u *Staging Area*, odnosno datoteka koje će biti deo sledećeg *commit*-a.
3. Izvršavanje *commit*-a, čime se pravi snimak izmenjenih datoteka navedenih u *Staging Area*.

Komande za inicijalizovanje repozitorijuma

Postoje dva načina za uspostavljanje *Git* repozitorijuma:

- **git init**

U okviru trenutnog direktorijuma, koji postaje radni, se kreira poddirektorijum *.git*, u kojem se nalaze sve potrebne datoteke za repozitorijum.

- **git clone <remote-url>**

Klonira se udaljeni repozitorijum koji se nalazi na datom linku u direktorijum koji će se nazvati po imenu udaljenog repozitorijuma. Kopira se ceo repozitorijum uključujući i *.git* direktorijum, što znači da se dobija pristup svakoj verziji svake datoteke u istoriji ovog repozitorijuma.

Komande za rad sa indeksnom datotekom

- **git add <path1> [<path2> <path3> …]**

Datoteka koja se nalazi na zadatoj putanji (ili datoteke na zadatim putanjama) se dodaje u indeksnu datoteku.

- **git add .**

Sve datoteke u radnom direktorijumu se dodaju u indeksnu datoteku.

- **git add –u**

Sve datoteke koje su izmenjene ili obrisane se dodaju u indeksnu datoteku, isključujući nove datoteke. Korisno kada ne želimo da dodajemo datoteke koje nisu bili praćene prethodno.

- **git add *.txt**

Jedan način korišćenja *wildcard*-a. Dodaje sve datoteke koje imaju ekstenziju .txt u indeksnu datoteku.

Komande za rad sa indeksnom datotekom

- **git reset [<path>]**

Brišu se sve datoteke iz indeksne datoteke. Ako se navede putanja do datoteke, samo ona se uklanja iz indeksne datoteke.

- **git status**

Prikazuje razlike između radnog direktorijuma i lokalnog repozitorijuma, kao i koje datoteke se nalaze u indeksnoj datoteci, a koje ne.

- **git diff [--staged]**

Prikazuje razlike između datoteka koje se nalaze u radnom direktorijumu i datoteka koje se nalaze u indeksnoj datoteci. Ako se navede opcija *staged*, prikazuje razlike između datoteka koje se nalaze u indeksnoj i njihovih verzija u poslednjem *commit-u.*

- **git restore <path> [--staged]**

Datoteka na putanji *path* se vraća na verziju u kojoj je bila u poslednjem *commit*-u. Opcija *staged* se navodi ako se datoteka nalazi u indeksnoj datoteci.

- **git rm <path>**

Datoteka na putanji *path* se briše iz radnog direktorijuma i indeksnog fajla.

Komande za rad sa lokalnim repozitorijumom

- **git commit -m “<message>“**

Snima izmene svih datoteka koje se nalaze u indeksnoj, i čuva snimak u bazi lokalnog repozitorijuma.

- **git commit -m “<message>“ –a**

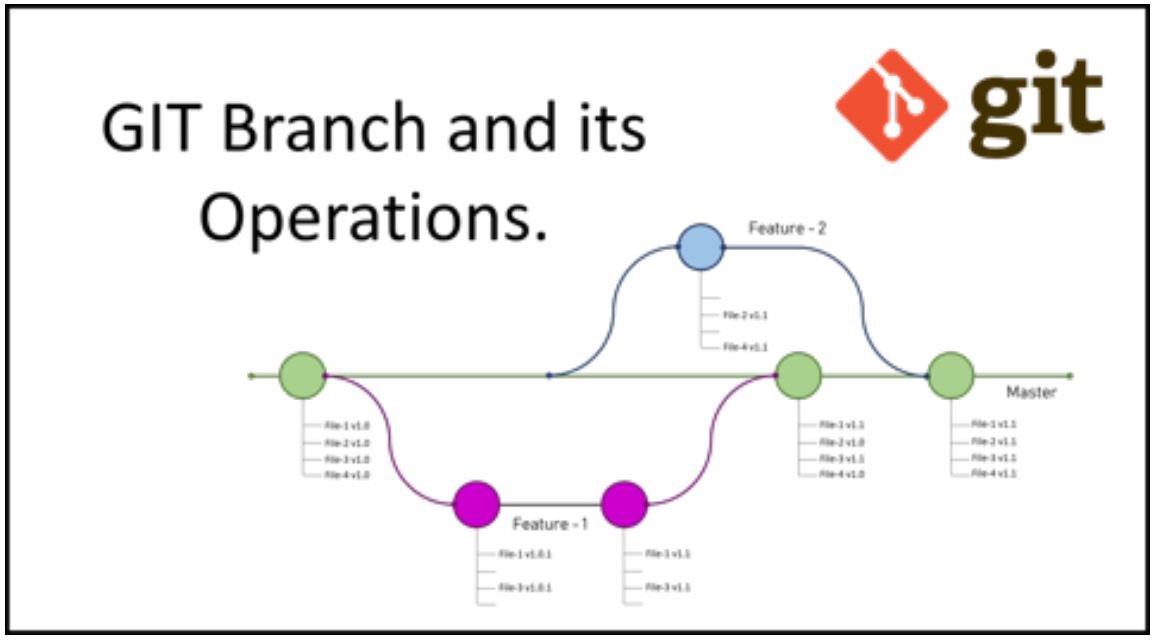
Stavlja sve datoteke koje su izmenjene ili izbrisane automatski u indeksnu datoteku, i zatim radi isti posao kao i bez opcije *a.*

- **git log [-p]**

Prikazuje istoriju svih *commit*-ova, uz opciju *p* prikazuje i izmene napravljene u svakom.

- **git revert <commit-hash>**

Vraća nazad izmene napravljene u *commit*-u identifikovanom argumentom *commit-hash*.



Grananje

Grananje je jedan od najbitnijih mehanizama u distribuiranim sistemima za kontrolu verzija kao što je *Git*. Grane dopuštaju odvajanje od glavne baze koda, dopuštajući programerima da rade na novim funkcionalnostima, ispravljaju greške ili eksperimentišu bez rizika uticanja na glavnu bazu koda dok nova izmena nije spremna. Grane omogućavaju da istovremeno više članova tima radi nezavisno na svojim dodacima na kod, i olakšavaju pregled koda dopuštajući članovima tima da pregledaju promene izolovano od ostatka baze koda.

Komande za rad sa granama

- **git branch <branch-name> [-d]**

Pravi novu granu sa imenom *branch-name*. Sa uključenom opcijom *d* briše datu granu.

- **git checkout <branch-name> [-b]**

Prebacivanje na granu sa imenom *branch-name*, dok se sa uključenom opcijom *b* ta grana napravi pre prebacivanja.

- **git merge <branch-name>**

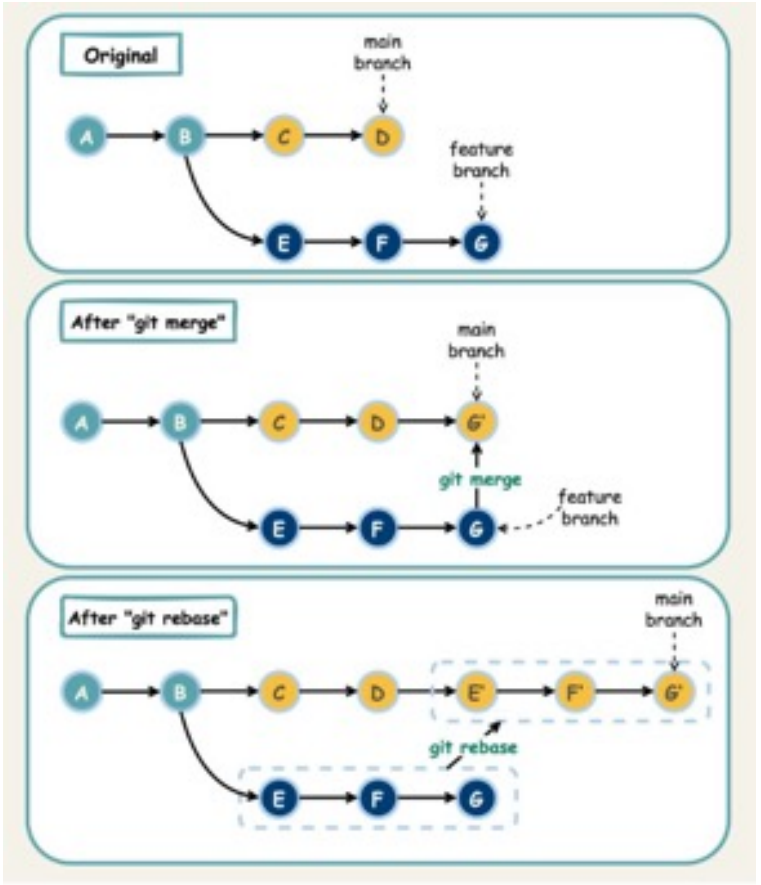
Spaja izmene napravljene na grani *branch-name* sa trenutnom granom, pri čemu može nastati **merge conflict!**

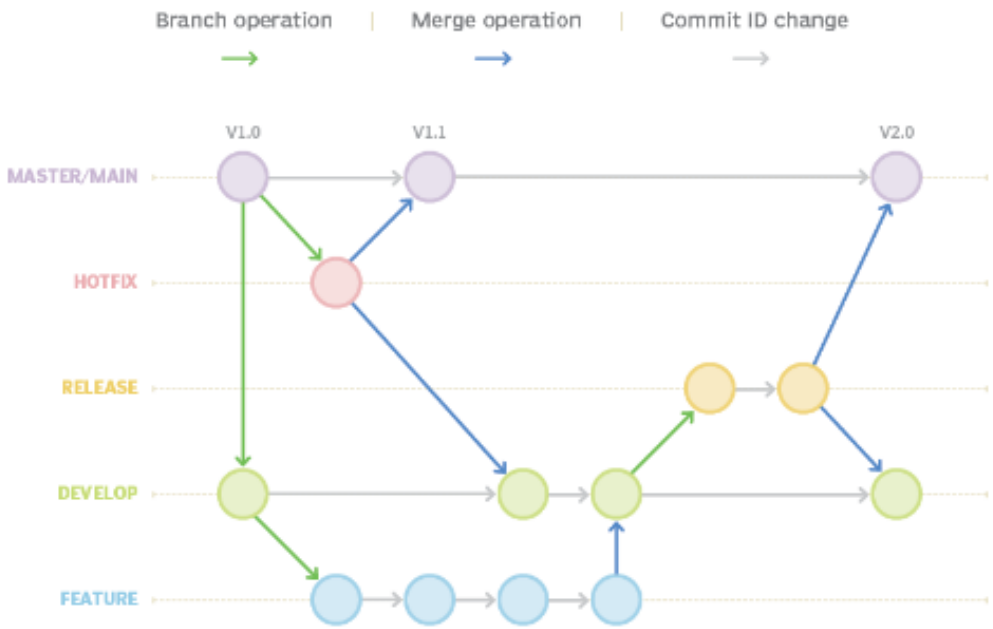
- **git stash, git stash apply**

Pri pozivu *stash,* čuvaju se izmene napravljene na trenutnoj grani, koje se mogu primeniti na drugoj pozivom *stash apply.*

- **git rebase <branch-name>**

Primenjuje *commit*-ove trenutne grane na vrh grane *branch-name*.





Gitflow

Gitflow predstavlja skup konvencija za organizovanje grana u *Git* repozitorijumima, dizajniran da olakša kolaborativni i organizovani razvoj na projektima. Grane su podeljene na sledeći način:

- *master* - na ovoj grani se nalazi kod spreman za produkciju i smatra se stabilnim.
- *develop* - grana koja se koristi za integraciju *feature* grana za testiranje pre prebacivanja na produkciju.
- *feature* - grane koje se koriste za razvijanje novih funkcionalnosti.
- *release* - grana na kojoj se priprema novo izdanje softvera.
- *hotfix* - grana na kojoj se prave brze ispravke, odvaja se od *master* grane da bi se, kad se ispravi mali *bug*, spojila nazad sa *master* i *develop* granama.

Popularni GUI alati za *Git:*

GitHub Desktop - https://desktop.github.com/

Sourcetree - https://www.sourcetreeapp.com/

GitKraken - https://www.gitkraken.com/

Komande za rad sa udaljenim repozitorijumom

- ## git remote add <remote_name> <repository_url>

Dodaje, tj. povezuje trenutni direktorijum sa udaljenim repozitorijumom na zadatom URL-u I imenuje taj repozitorijum po imenu datom prvim argumentom.

Komanda **git clone** implicitno izvršava ovu komandu posle kopiranje udaljenog repozitorijuma, I naziva ga **origin**.

- ## git remote remove <remote_name>

Uklanja vezu ka udaljenom repozitorijumu sa imenom *remote_name*.

- ## git branch -r

Ispisuje sve grane koje se nalaze na udaljenom repozitorijumu.

Komande za rad sa udaljenim repozitorijumom

- **git fetch <remote> <remote_branch>:<local_branch>**

Skida metapodatke i datoteke sa grane date argumentom *remote_branch* koja se asocira sa granom datom argumentom *local_branch* sa udaljenog repozitorijuma definisan imenom *remote.*

- **git pull <remote> [--rebase]**

Izvršava **fetch** komandu sa tekuće grane za udaljeni repozitorijum definisan imenom *remote*, a zatim izvršava **merge** nad lokalnom granom (može se desiti **merge conflict!**). Sa uključenom opcijom --rebase umesto merge-a se izvršava rebase.

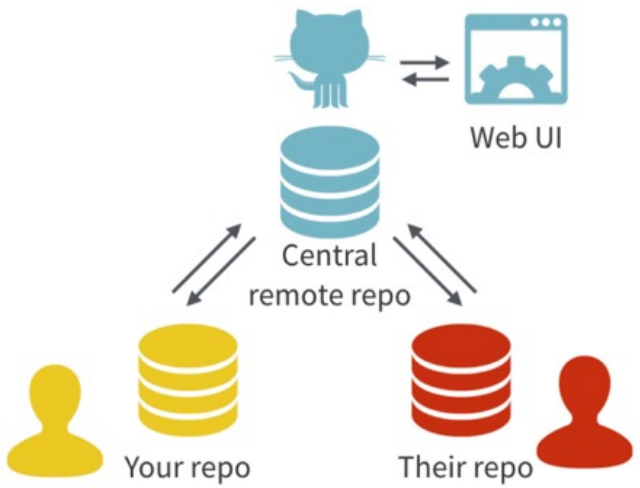
- **git push <remote> <branch>**

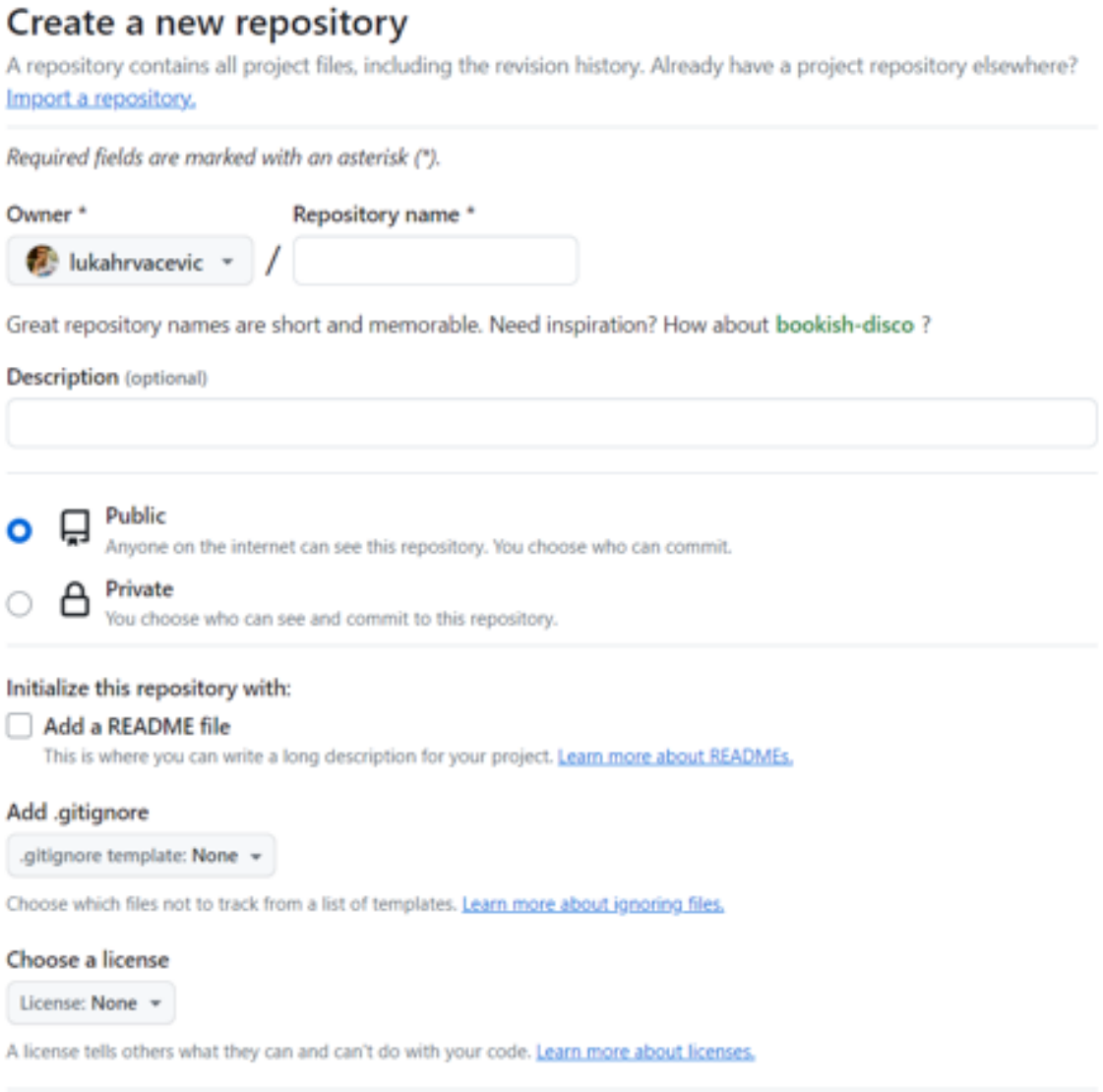
Na udaljeni repozitorijum sa imenom *remote* gura sve izmene koje se nalaze na lokalnoj grani, a ne nalaze na udaljenoj grani sa nazivom *branch*.

*Savet: Uvek prvo izvršiti komandu **pull**, pa zatim **push*** (eventualno pre komande **pull** izvršiti **stash**)

GitHub

- *Git* – distribuirani sistem za kontrolu verzija, *GitHub* – veb aplikacija za hosting udaljenih *Git* repozitorijuma
- *GitHub* na neki način dopunjuje *Git* pružajući korisnički interfejs za neke komande i mehanizam za udaljene repozitorijume
- *GitHub* je kao *DropBox* ili *Google Drive*, ali više struktuiran, moćan i programabilan

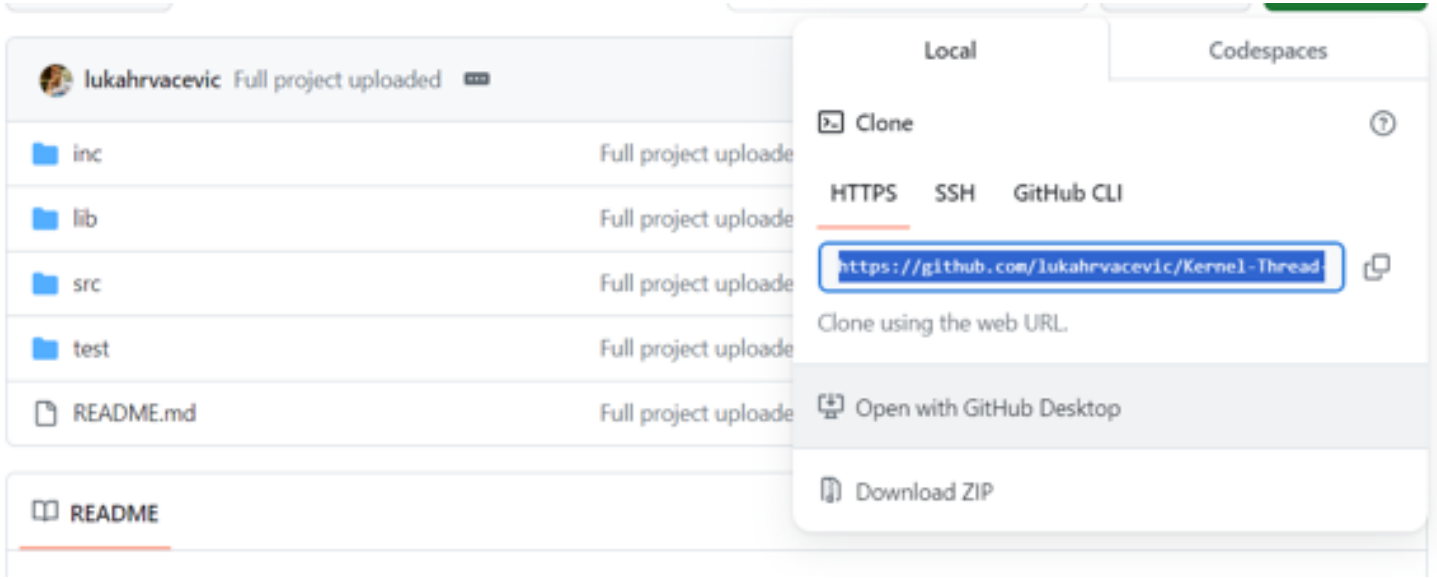




Kreiranje *GitHub* repozitorijuma

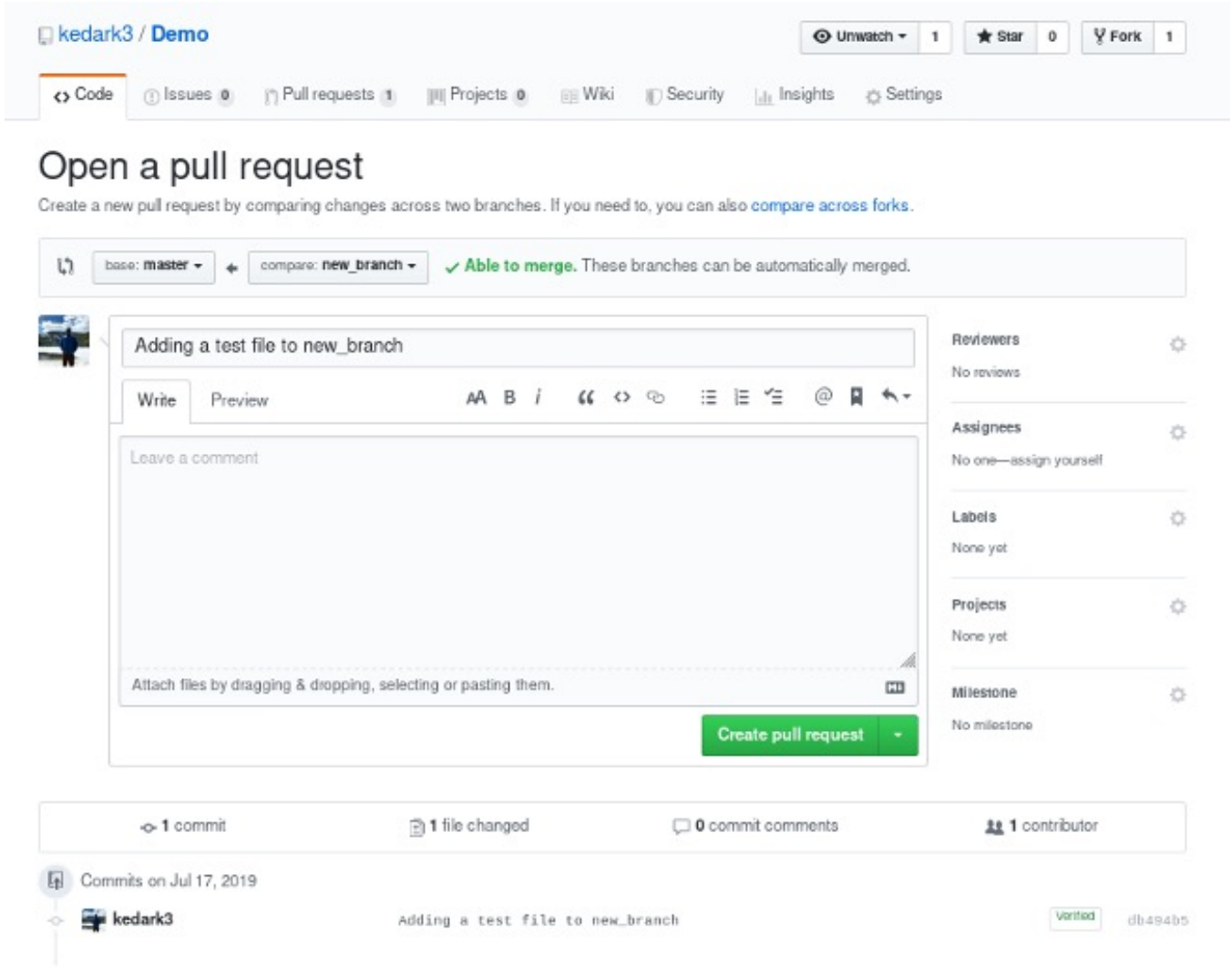
- *README file* – datoteka koja se piše u *markdown* sintaksi i treba da sadrži opis repozitorijuma. Prikazuje se na glavnoj stranici samog repozitorijuma
- *.gitignore file* – datoteka u kojoj se piše spisak datoteka koje *Git* treba da ignoriše
- Ako se napravi privatni repozitorijum, naknadno se mogu dodati kolaboratori koji će imati pristup datom repozitorijumu

Povezivanje sa udaljenim *GitHub* repozitorijumom



1. Otvoriti komandnu liniju u radnom direktorijumu
2. Izvršiti komandu –

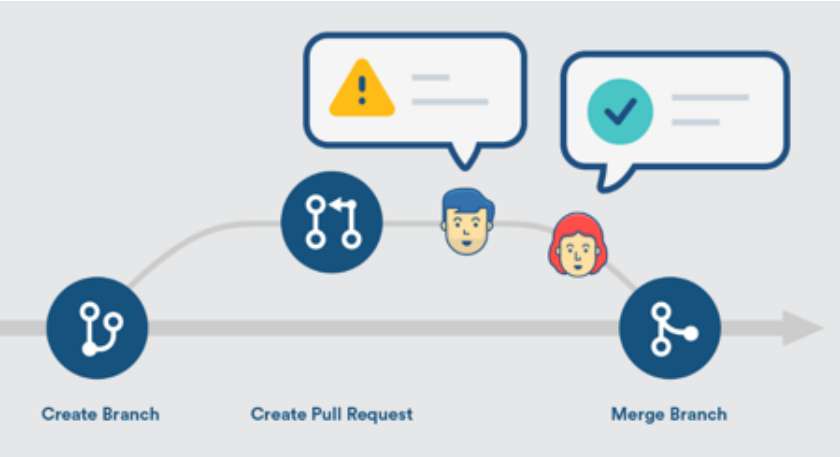
   git clone <repo-url>
3. Ako nije podešen nalog ranije, prompt će tražiti username i password za *GitHub* nalog



Pull request

Pull request je mehanizam implementiran na *GitHub-u*, koji se koristi za predlaganje integracije promena iz jedne grane u drugu. Omogućava *code review* od strane ostalih kolaboratora na repozitorijumu i komentarisanja/menjanja izmena

pre nego što se date izmene i integrišu.

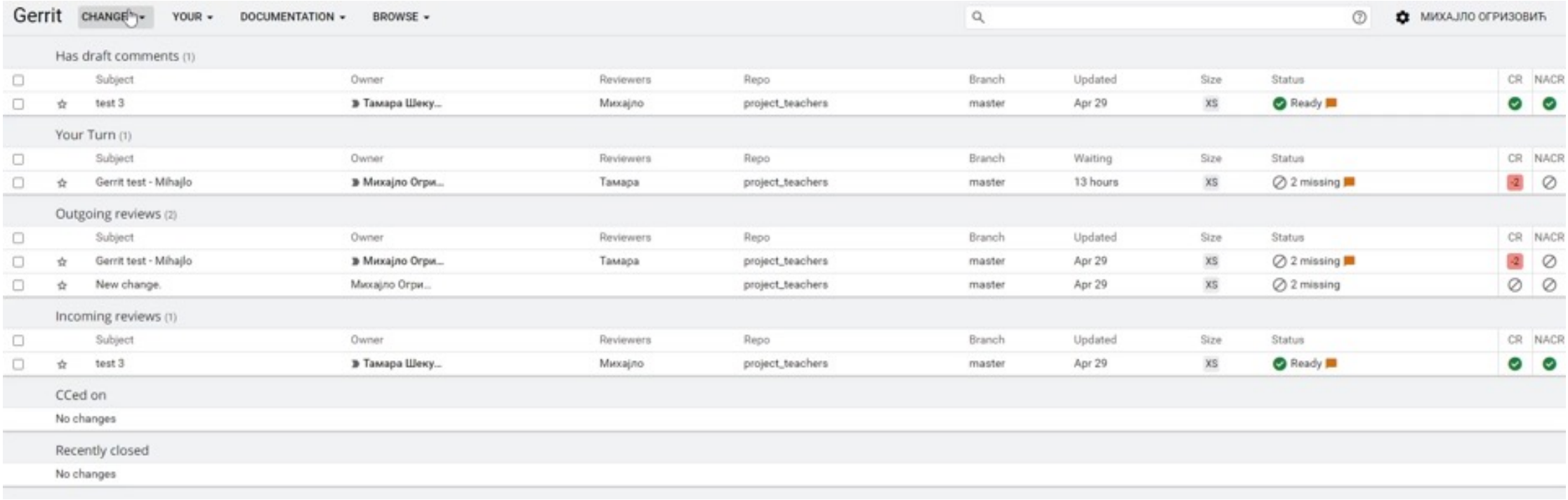


Koraci koji se obično prate kada se *pull request* mehanizam koristi su sledeći:

1. Član tima pravi novu granu na svom lokalnom repozitorijumu, na kojoj će praviti izmene ili dodavati nove funkcionalnosti.
2. Pravi potrebne izmene i verzioniše ih na svojoj grani.
3. Nakon odrađenog dela posla koji je bio zamišljen, lokalna grana se gura na udaljeni repozitorijum.
4. Pravi se *pull request*, čime se zahteva integracija novonapravljenih izmena na (glavnu) granu.
5. Ostali kolaboratori na repozitorijumu vrše *code review*, odnosno pregledaju izmene koje su napravljene pri čemu mogu ostaviti komentare kojima postavljaju pitanja, daju preporuke ili pružaju informacije od interesta.
6. Nakon dobijenih povratnih informacija, autor izmena eventualno pravi dodatne izmene.
7. Nakon što su izmene pregledane i odobrene, *pull request* se *merge*-uje, čime se izmene integrišu u datu granu.

Gerrit

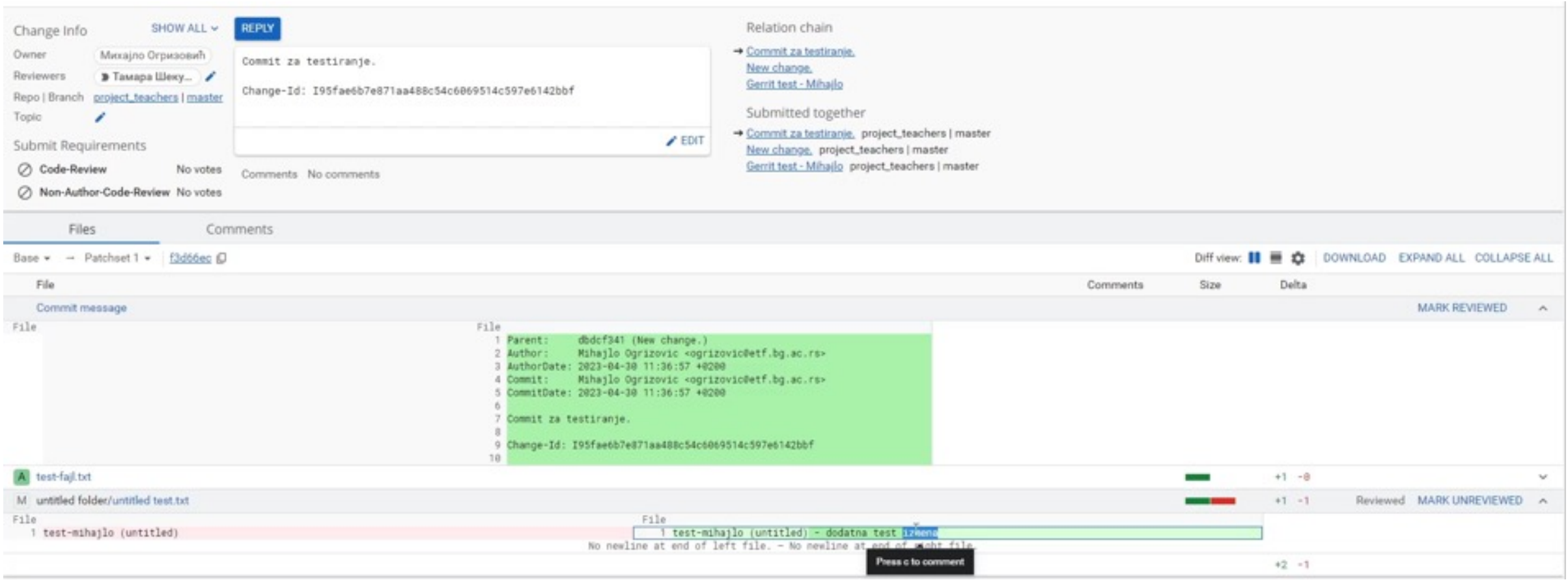
Gerrit je alat za upravljanje repozitorijumima i procesom *code review-a* koji se često koristi u razvoju softvera. Gerrit omogućava timovima da pregledaju, komentarišu i odobravaju ili odbijaju promene koje su predložene u kodu pre nego što se te promene integrišu u glavnu granu.



Gerrit je hostovan na sajtu https://crete.etf.bg.ac.rs/ gde će se nalaziti repozitorijumi napravljeni za vaše korišćenje tokom izrade projekta. Povezivanje na repozitorijum je opisano u dokumentaciji.

Pored *git clone* komande će biti potrebno i da se skine commit-msg hook, koji je potreban za interno funkcionisanje *Gerrit*-a, a koji na svaku *commit* poruku dodaje identifikator koji *Gerrit* koristi.

Na samom udaljenom repozitorijumu se nalazi samo grana master, na koju nije dozvoljeno guranje bilo kakvih promena. Guranje promena se vrši pomoću komande *git push origin HEAD:refs/for/master,* koja u suštini okida *code review* proces čineći ovu promenu vidljivom svim ostalim članovima tima koji su povezani na repozitorijum.



Kada ova promena postane vidljiva, ostali članovima tima mogu preko sajta pogledati šta je sve izmenjeno u okviru tog *commit*-a, komentarisati određene delove izmena i finalno ostaviti ocenu. Moguće ocene su:

- -2 ; veto na izmenu, izmena ne može nikako da prođe.
- -1 ; ne deluje dobro (ali ako neko drugi da +2 može da prođe).
- 0 ; neutralno
- +1 ; izmene su u redu (ali nek još neko pogleda).
- +2 ; izmene mogu da prođu.

Uslov da bi neka promena prošla jeste da je bar još jedan član tima ocenio izmenu, da postoji bar jedna +2 ocena, da ne postoji nijedna -2 ocena i da su svi *commit*-ovi koji su se desili pre datog već odobreni.

Ako se tokom *code review*-a zaključi da nešto u okviru te izmene mora da se promeni, ta promena se pravi nakon čega se poziva *git commit --amend* na lokalnom repozitorijumu i gura na udaljeni opet, čime to ostaje deo jednog *commit*-a.