

Java

Sistemi za kontrolu verzija

Sistemi za kontrolu verzija

- ▶ Sistemi za kontrolu verzija se koriste kada je potrebno pratiti verzije projekta i/ili kada više programera radi na projektu
- ▶ Bazirani su na repozitorijumima koda u koji se smeštaju fajlovi i folderi koji čine projekat

Sistemi za kontrolu verzija

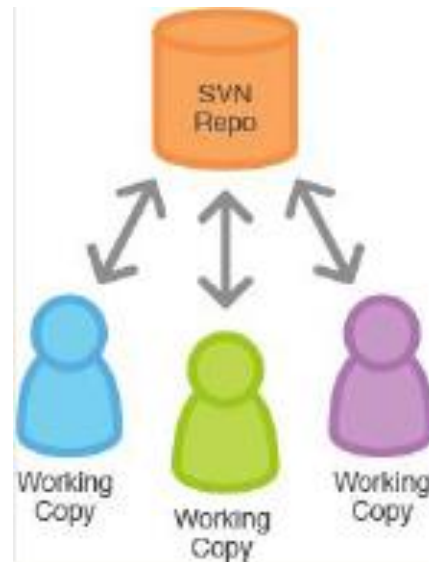
- ▶ Svaka izmena koja se sačuva na repozitorijumu (zvana commit) pored samih izmena čuva i informacije o tome ko je napravio izmenu, kad je izmena sačuvana i šta je tačno izmena
- ▶ Dostupan je istorijat izmena projekta na repozitorijumu, te je moguće pristupiti ranijim verzijama projekta i svim relevantnim podacima

Sistemi za kontrolu verzija

- ▶ Dva osnovna tipa sistema za kontrolu verzija
 - Centralizovani
 - Distribuirani (Git, Mercurial)

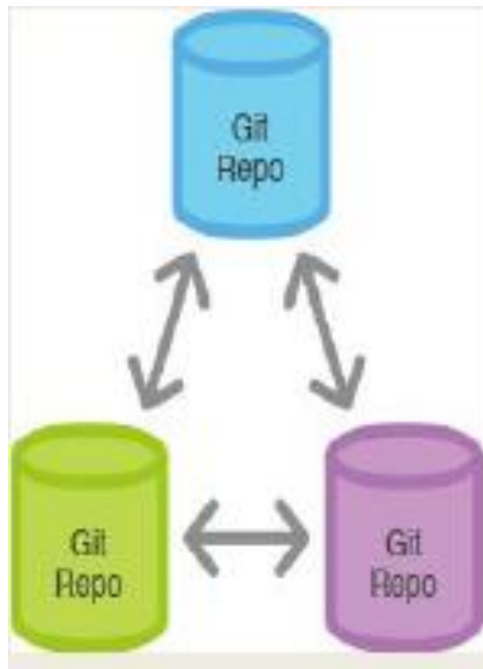
Centralizovani sistemi za kontrolu verzija

- ▶ glavni repozitorijum se nalazi na centralnom serveru, dok ostali čvorovi imaju samo preuzetu verziju projekta
- ▶ Najpopularniji predstavnici su SVN i CVS

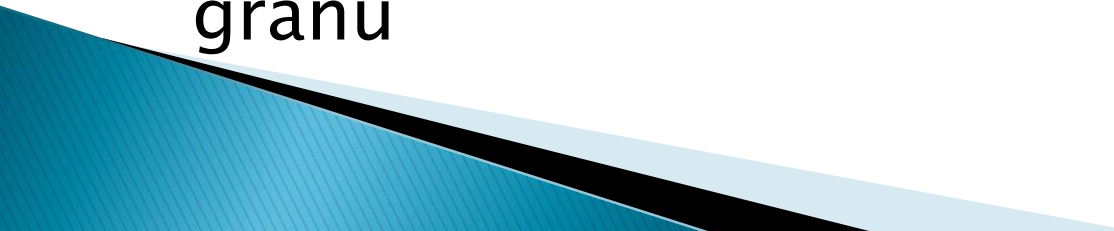


Distribuirani sistemi za kontrolu verzija

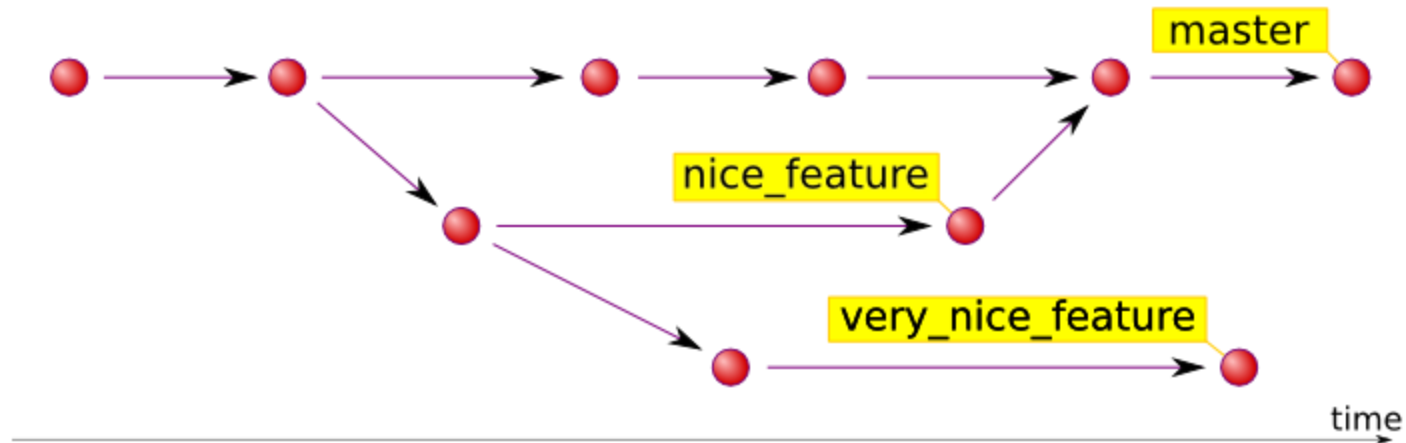
- ▶ Najpopularniji predstavnik je Git
- ▶ Svaki čvor predstavlja ravnopravan repozitorijum sa kompletnom istorijom izmena



Git repozitorijum

- ▶ Sadrži fajlove i foldere sa kompletnom istorijom izmene
 - ▶ Repozitorijum čuva sekvencu izmena
 - ▶ Moguće je formirati više grana u repozitorijumu tako da izmene projekta na jednoj grani ne utiču na ostale grane, što pruža podršku eksperimentisanju i nezavisnom razvoju delova sistema
 - ▶ Svaka grana ima sekvencu izmena
 - ▶ Moguće je odvojiti novu granu iz postojeće
 - ▶ Grana se može ponovo spojiti u postojeću granu
- 

Git repozitorijum –izmene i grane



* Preuzeto sa hades.github.io

Git repozitorijum –izmene i grane

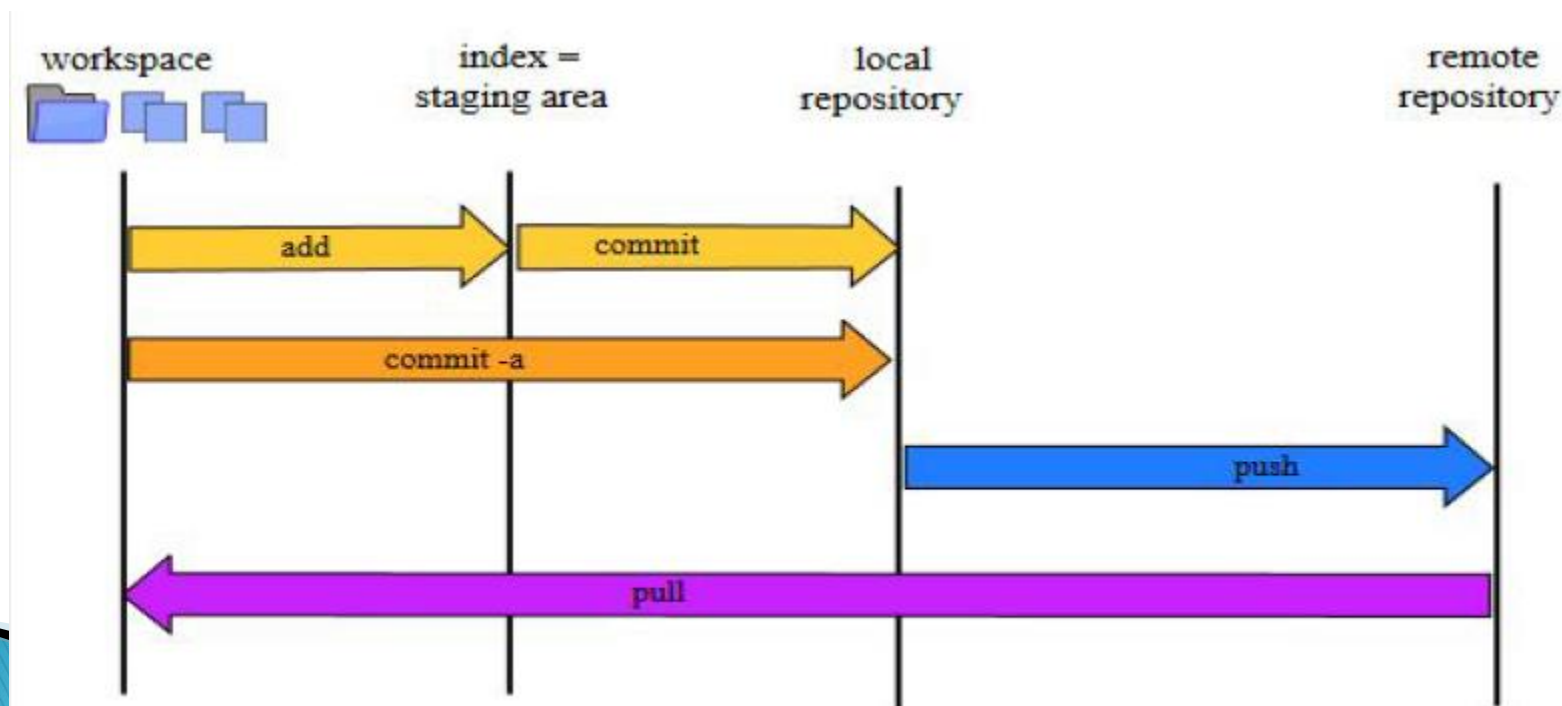
- ▶ Glavna grana se zove master
- ▶ Možemo pristupati bilo kojoj izmeni od ranije
- ▶ Fajl sistem u jednom trenutku može da odražava sadržaj iz bilo koje izmene
 - Ta izmena na koju smo trenutno „pozicionirani“ je označena kao HEAD
 - Ako smo trenutno na najnovijoj verziji projekta master grane, onda HEAD pokazuje na poslednji commit u master grani

Git repozitorijum

- ▶ Kreiranje novog lokalnog Git repozitorijuma u postojećem direktorijumu
 - `git init`
- ▶ Kloniranje postojećeg repozitorijuma
 - `git clone <putanja do repozitorijuma> <lokalni folder u kojem ce biti repozitorijum kreiran>`
 - `git clone https://github.com/libgit2/libgit2 temp`
 - ako server ima *self-signed* sertifikat
 - `git -c http.sslVerify=false clone https://puppet.ftn.uns.ac.rs/2015z/jwts/tim1/git tim1_git_repo`

Postavljanje izmena na repozitorijum

- ▶ Postavljanje izmena se vrši u dve faze
 - postavljanje na *staging area*, gde se commit može pripremiti i
 - sam commit



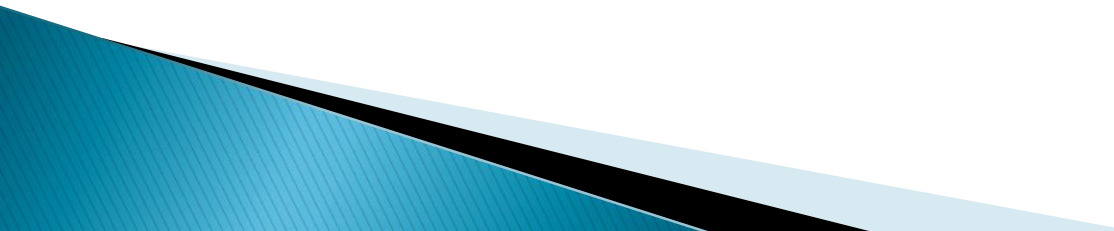
Fajlovi u *staging area*

- ▶ Postavljanje jednog fajla
 - `git add nekiFajl.txt`
- ▶ Postavljanje svih fajlova u repozitorijumu
 - `git add .`
- ▶ Brisanje fajla iz fajl sistema i postavljanje te izmene u staging area
 - `git rm nekiFajl.txt`
- ▶ Poništavanje postavljanja
 - `git reset HEAD nekiFajl.txt`

Postavljanje na repozitorijum

- ▶ Operacija commit
 - prebacuje izmene iz staging area na repozitorijum
 - commit -m "Opis izmene."

Zašto se postavljanje izmena vrši u dve faze?

- ▶ Korisno za logičko odvajanje izmena u više commit-a
 - ▶ Jedan commit treba biti moguće opisati kroz rečenicu ili dve
 - ▶ Težiti ka davanju prefiksa opisa commit-a (Add, Fix, itd.) kako bi se ukazalo na tip izmene
- 

Ignorisanje fajlova

- ▶ Git može automatski da ignoriše određene fajlove koje ne želimo da čuvamo na repozitorijumu
- ▶ Npr. kompajlirane fajlove nema potrebe da čuvamo na repozitorijumu, dovoljni su izvorni
- ▶ Kreira se u repozitorijumu fajl `.gitignore`
- ▶ U fajlu se navedu fajlovi/šabloni fajlova koji treba da budu ignorisani, npr.
 - `.class`
 - `.o`

Stanje repozitorijuma

- ▶ git status
- ▶ Prikazuje izmene koje
 - treba da budu postavljene u staging area
 - su postavljene u staging area, ali nisu na repozitorijum

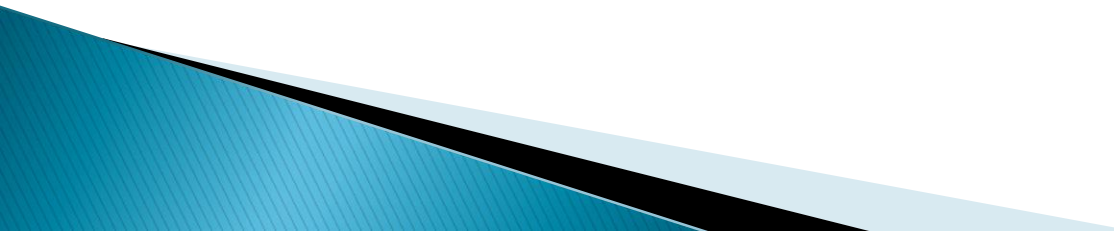
Istorija repozitorijuma

- ▶ git log
- ▶ Prikazuje sve commit-ove sa opisom, autorima itd.
- ▶ Svaka izmena ima svoj jedinstven identifikator (hash)
- ▶ gitk
 - grafički alat sa prikazom istorije repozitorijuma

Preuzimanje ranije izmene

- ▶ Repozitorijum se može vratiti u stanje u kojem je bio u određenoj izmeni
- ▶ `git checkout <hash>`
 - Najčešće je već prvih 7 karaktera hash koda jedinstveno, pa je dovoljno samo njih uneti
- ▶ Sada HEAD „pokazuje“ na tu izmenu koju smo preuzeli

Rad sa granama

- ▶ Omogućavaju izolovan rad na različitim komponentama sistema i pružaju dobru podršku testiranja ideja
 - ▶ Razvijamo komponentu i kad smo zaokružili celinu spajamo (merge) kod komponente sa glavnim granom
 - ▶ Preporuka je da master grana ima samo merge, a ne commit čvorove
 - ▶ HEAD pokazivač pokazuje na kojoj grani i čvoru smo trenutno pozicionirani
- 

Rad sa granama

- ▶ Kreiranje grane
 - `git branch <ime grane>`
- ▶ Prebacivanje na granu
 - `git checkout <ime grane>`
 - pomera HEAD pokazivač na datu granu i menja sadržaj fajl sistema tako da oslika stanje repozitorijuma u datom čvoru
 - Neposredno nakon kreiranja grane, sadržaj fajlova je isti kao i na grani sa koje smo se odvojili

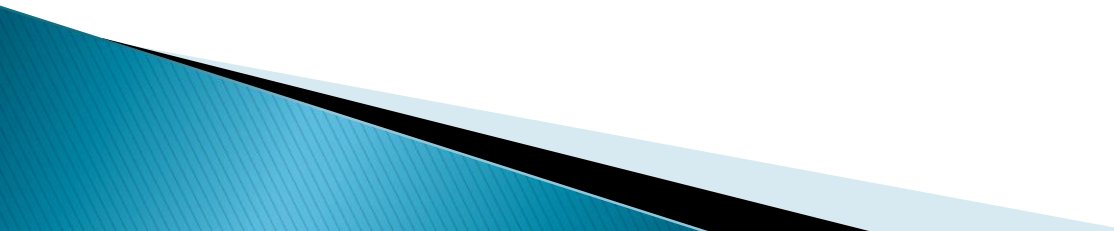
Rad sa granama

- ▶ Prikaz svih grana
 - git branch
 - U spisku prikazanih grana, stoji * pored imena grane na koju smo trenutno pozicionirani

Spajanje izmena u različitim granama – merge

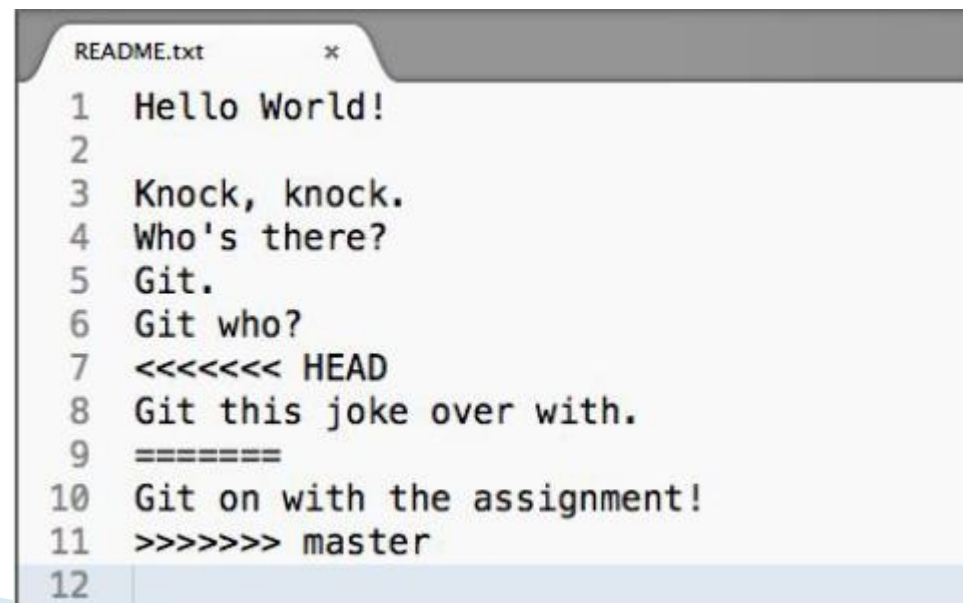
- ▶ Moguće je u jednu granu ubaciti i izmene napravljene u drugoj
- ▶ git merge <druga grana>
 - U granu na kojoj smo trenutno pozicionirani ubacuje i izmene napravljene na grani <druga grana>
 - Izmene se “spajaju”
 - Novi čvor će sadržati izmene napravljene i na prvoj i na drugoj grani

Konflikti

- ▶ Spajanje izmena će biti uspešno izvršeno samo ako u granama koje se spajaju ne postoje izmene u istim linijama istih fajlova
 - ▶ Ako ovo nije slučaj, fajl čiji sadržaj nije mogao biti „spojen“ je u konfliktu
 - ▶ U fajlu koji je u konfliktu označene su konfliktne linije
- 

Konflikti

- ▶ Obe grane su promenile liniju 7
- ▶ Na mestu linije 7, Git je izgenerisao kod sa
 - Sadržajem trenutne grane (označeno markerom HEAD) – na slici linija 8
 - Sadržajem grane sa kojom se spajamo (označeno imenom te grane) – na slici linija 10



```
README.txt  ✕
1  Hello World!
2
3  Knock, knock.
4  Who's there?
5  Git.
6  Git who?
7  <<<<<<< HEAD
8  Git this joke over with.
9  =====
10 Git on with the assignment!
11 >>>>>>> master
12
```


Konflikti

- ▶ Potrebno je srediti sadržaj konfliktnog fajla tako da se obrišu markeri i neželjeni sadržaj
- ▶ Nakon što se uredi sadržaj fajla potrebno je dodati fajl na *staging area* putem git add, što razrešava konflikt

Rad sa udaljenim repozitorijumom

- ▶ Iako su svi repozitorijumi u Gitu ravnopravni, najčešći scenario za timski rad je da postoji jedan javno dostupan repozitorijum koji čuva izmene svih članova tima
- ▶ Ovaj *default* udaljeni repozitorijum se zove *origin*
- ▶ Pri kloniranju repozitorijuma, za *origin* se postavlja repozitorijum čiji se sadržaj klonira

Slanje izmena na udaljeni repozitorijum

- ▶ Slanje sadržaja lokalnog repozitorijuma na udaljeni repozitorijum
 - `git push <ime ili url repozitorijuma> <ime grane koju šaljemo>`
 - `git push origin master`

Preuzimanje izmena sa udaljenog repozitorijuma

- ▶ Preuzimanje izmena sa udaljenog repozitorijuma
 - `git fetch <ime ili url repozitorijuma>`
 - `git fetch origin`
 - Isto radi i `git fetch` (jer je origin *default*)
- ▶ Preuzete izmene je potrebno onda spojiti sa lokalnom granom
 - `git merge origin/master`
 - ako se spajamo sa granom master sa udaljenog repozitorijuma

Preuzimanje izmena sa udaljenog repozitorijuma

- ▶ Prethodne dve naredbe se mogu objediniti
- ▶ `git pull <naziv ili url repozitorijuma>`
 - Radi fetch, pa zatim merge

Konflikti pri preuzimanju izmena iz lokalnog repozitorijuma

- ▶ Ako dva korisnika inicijalno povuku istu verziju datoteke (git pull), nakon čega oba korisnika izmene datoteku i sačuvaju izmene u lokalni repozitorijum
- ▶ Oba korisnika izvrše git push naredbu da pošalju izmene na udaljeni repozitorijum:
 - prvi push prolazi bez problema i čuva se izmena na repozitorijumu
 - drugi push zahteva od korisnika da preuzme najnoviju verziju udaljenog repozitorijuma pre nego što može da izvrši push;
 - ukoliko u ovom trenutku izvrši operaciju pull, drugi korisnik dobija konflikt
 - konflikt se razrešava na isti način kao kod merge-a lokalnih grana
- ▶ preporuka je raditi git fetch i onda git merge kako bi imali veću kontrolu nad procesom i veću svest o tome šta se zapravo izmenilo