Napredni algoritmi i strukture podataka

Sistemi za kontrolu verzija, Git, GitFlow

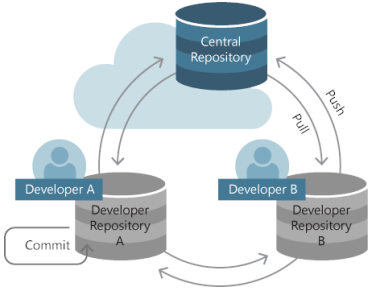
Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Version Control Systems (VCS)

- Programi koji se koriste kada je potrebno pratiti verzije projekta i/ili kada više učesnika radi na projektu
- Danas su nezaobilazni u procesu razvoja softvera
- Datoteke koje se prate zajedno sa svom istorijom promena se nalaze na udaljenom mestu — **repozitorijum**
- Svaka izmena koja se uradi, treba da se sačuva na repozitorijumu — **commit**
- Pored samih izmena čuva i informacije o tome ko je napravio izmenu, kad je izmena napravljena i šta je tačno izmenjeno
- Dva osnovna tipa sistema za kontrolu verzija:
  - **Centralizovani** (Subversion – SVN, CVS, ...)
  - **Distribuirani** (Git, Mercurial...)

Git

- Preuzeti instalaciju sa službene web stranice
- Prvobitna namena je korišćenje iz komandne linije (grafička okruženja za Git dostupna su na linku)



Podešavanja

- Nakon instalacije, dobra praksa je da se odmah podesi ime i email adresa koje će Git koristiti prilikom čuvanja izmena:

```
git config --global user.name "Ime Prezime"
git config --global user.email mailadresa@nesto.com
```

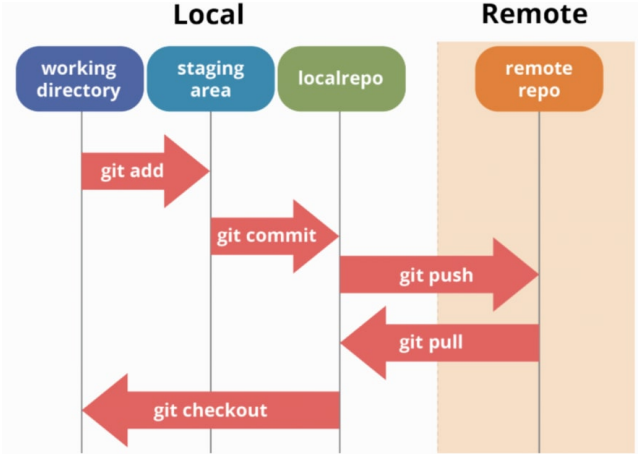
- *global* će izvršiti podešavanja za svaki repozitorijum na računaru
- Ukoliko se izostavi, podešavanja se vrše samo za onaj repozitorijum na koji smo trenutno pozicionirani iz komandne linije
- lista trenutno podešenih vrednosti se može dobiti komandom:

```
git config --list
```

Kreiranje lokalnog repozitorijuma

- Postoje dva načina:
  1. **Kloniranjem postojećeg repozitorijuma sa udeljanog servera**
     - *git clone [putanja_do_repozitorijuma] [lokalni_direktorijum_gde_se_klonira]* je komanda koja će kreirati lokalni repozitorijum sa svim fajlovima i istorijom izmena sa udeljenog servera
     - U okviru radnog direktorijuma kreira skriveni folder **.git** koji zapravo označava da je folder git repozitorijum
  2. **Inicijalizacijom novog (praznog) repozitorijuma**
     - *git init* je komanda koja će kreirati prazan lokalni repozitorijum
     - Kreiraće se sakriveni folder **.git** unutar direktorijuma iz kojeg smo pokrenuli komandu (radnog direktorijuma)
     - Treba da podesimo adresu udaljenog **origin** repozitorijuma
     - *git remote add origin [adresa_repozitorijuma]*

Git file status lifecycle



Praćenje datoteka

- Kreiranje datoteka u radnom direktorijumu neće navesti Git da prati izmene nad njima
- Pokretanje praćenja verzija nad nekom datotekom ili direktorijumom se vrši komandom *git add [naziv_datoteke]*
- Komanda *git add* se mora izvršiti za sve datoteke koje su nove, ali i za sve postojeće datoteke nad kojima su izvršene neke izmene (nalaze se u stanju modified)
- Možemo automatski dodati sve izmene komandom *git add .*
- Naknadne izmene nad praćenim datotekama neće biti evidentirane automatski
- Prilikom svake izmene potrebno je ponoviti *git add* komandu

Postavljanje izmena na repozitorijum

- Vrši se u dve faze:
  1. postavljanje promena u deo repozitorijuma koji se zove **staging area (evidentiranje izmena)** — datoteke se u ovaj deo repozitorijuma dodaju komandom *git add* (za objekte koji pripadaju ovom delu se kaže da su u staged stanju)
  2. "pakovanje" evidentiranih izmena u **commit** — predstavlja prebacivanje izmene iz staging area na trenutnu granu repozitorijuma (format naredbe: *git commit -m "opis izmena"*)
- Saveti oko commit-a
  - Logički nezavisne izmene organizovati u odvojenim *commit-ima*
  - Jedan *commit* treba biti moguće opisati kroz rečenicu ili dve
  - Težiti ka davanju prefiksa opisu *commit-a* (add,fix,itd.) kako bi se ukazalo na tip izmene

Postavljanje izmena na repozitorijum

- Operacija *commit* izmene šalje na lokalni repozitorijum
- Operacija *push* šalje lokalne izmene na udaljeni repozitorijum (može više commit-a)
- Uobičajeni koraci:

```
git add .
git commit -m "Tekst commit poruke"
git push origin master
```

Ignorisanje datoteka

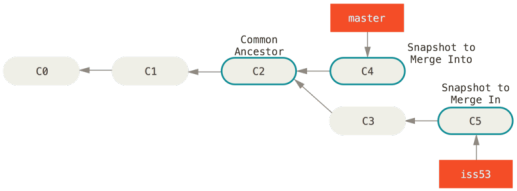
- Pokretanje *git add* komande nad celim direktorijumima može dovesti do toga da u praćenim datotekama budu i neke koje ne želimo da imamo na repozitorijumu (npr. kompajlirani fajlovi nisu potrebni, dovoljno je da imamo source fajlove, logovi, biblioteke...)
- Git omogućava način da navedemo pravila za ignorisanje tih datoteka ili direktorijuma prilikom pokretanja komande *git add*
- Pravila se navode u datoteci sa imenom *.gitignore* koji se kreira u repozitorijumu

Stanje repozitorijuma

- Stanje repozitorijuma možemo dobiti komadom *git status*
- Prikazuje izmene koje:
  - treba da budu postavljene u staging area
  - su postavljene u staging area, ali nisu na repozitorijum
- Prikazuje i informacije kao što su koja je trenutno aktivna grana i broj *commit-a* koji nisu poslati na udaljeni repozitorijum

Rad sa granama

- Omogućavaju izolovan rad na različitim komponentama sistema i pružaju dobru podršku testiranju ideja
- Novokreirani repozitorijumi imaju samo jednu granu koja se zove **master (main)**
- **HEAD** pokazivač pokazuje na kojoj grani i čvoru smo trenutno pozicionirani
- Razvijamo komponentu i kad smo zaokružili celinu spajamo **(merge)** kod komponente sa glavnom granom



Komande

- Kreiranje grane: *git branch [ime_grane]*
- Pozicioniranje na granu: *git checkout [ime_grane]*
- Prikaz svih grana: *git branch -a*
- Brisanje grane: *git branch –d [ime_grane]*

Spajanje grana

- Prilikom spajanja, potrebno je vratiti se na roditeljsku granu i pozvati komandu *git merge [druga_grana]*
  - U granu na kojoj smo trenutno pozicionirani ubacuje se izmene napravljene na grani [druga_grana]
  - Izmene se "spajaju" tako da novi čvor sadrži izmene napravljene i na prvoj i na drugoj grani
- Drugi način spajanja je *git rebase*
  - Ne kreira novi commit
  - Uglavnom služi u svrhe dopunjavanja grana izmenama sa master grane

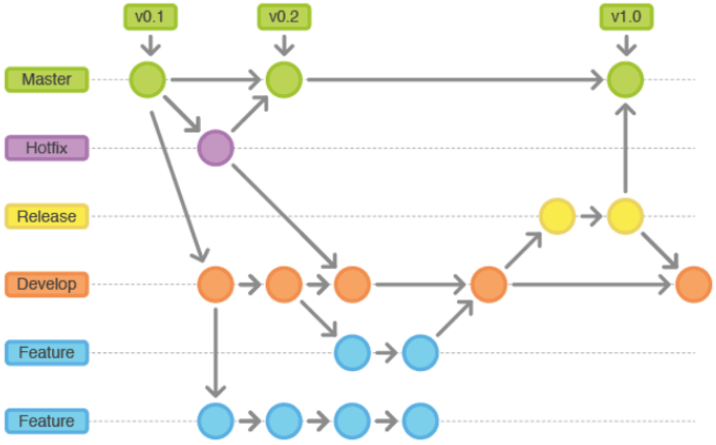
Konflikti

- Automatsko spajanje izmena će biti uspešno izvršeno samo ako u granama koje se spajaju ne postoje izmene u istim linijama istih datotetka
- Ako ovo nije slučaj, datoteka čiji sadržaj nije mogao biti "spojen" je u konfliktu
- U datoteci koja je u konfliktu označene su konfliktne linije
- Primer konflikta

```
<<<<<<< HEAD:index.html
contact : email.support@github.com
=======
please contact us at support@github.com
>>>>>>> develop:index.html
```

- Potrebno je srediti sadržaj konfliktne datoteke tako da se obrišu oznake i neželjeni sadržaj, a zatim komadom *git add* dodati datoteku na staging area

GitFlow model



(Image from https://anarsolutions.com/gitflow-branching-model/)

Dodatni materijali

- Git docs
- Git internals
- Gitflow
- Git tutorial

Zadaci I

- Podelite se u timove od po 2-3 člana
- Napravite nalog na GitHub-u
- Jedan član tima treba da kreira repozitorijum na GitHub-u i doda ostale članove kao saradnike (Settings → Manage access → Add people)
- Kada se svi pridruže repozitorijumu, treba da ga kloniraju lokalno kod sebe
- Podesite ime i email adresu koju će git koristiti prilikom čuvanja izmena u ovom repozitorijumu
- Jedan član tima treba da napiše main.go fajl koji u main funkciji ispisuje njegovo ili njeno ime i prezime na konzoli, doda .gitignore fajl i pošalje izmene na udaljeni repozitorijum
- Ostali članovi treba da preuzmu te izmene

Zadaci II

- Svaki član treba da napravi po jedan novi go fajl u kom će da napiše funkciju koja vrši jednu od ponudjenih operacija: sabiranje, oduzimanje, množenje, deljenje itd.
- Funkcije ne razvijate direktno na glavnoj grani, već svako na svojoj feature grani
- Kada završite implementaciju funkcionalnosti i pošaljete je na udaljeni repozitorijum, treba da je spojite sa glavnom granom, tako da na kraju svi članovi tima lokalno na glavnoj grani imaju pristup svim novim funkcijama
- Svi članovi tima treba da izmene main funkciju tako što će da umesto trenutnog imena ispišu svoje ime na konzoli, a ispod toga pozovu funkciju koju su implementirali (ovo radite na glavnoj grani, a zadatak je završen kada main funkcija ima pozive svih funkcija, a na konzoli se ispisuje ime poslednje osobe koja je komitovala izmene)