Git & code conventions

Repository

- Može se posmatrati kao direktorijum koji čuva sve fajlove, foldere i ostali sadržaj potreban jednom projektu
- Pored samog sadržaja projekta, čuva verzije svih datoteka, istoriju promena, komite, itd.
- Dva tipa: privatni i javni
- Korisnici koji imaju pristup repozitorijumu su organizovani po rolama i tako se definiše šta im je dozvoljeno a šta nije

Clone

- Clone predstavlja kopiju repozitorijuma, odnosno akciju sa kojom se određeni repozitorijum kopira
- Kloniranje lokalnog repozitorijuma
  - **git clone /path/to/repository**
- Kloniranje remote repozitorijuma
  - **git clone username@host:/path/to/repository**

Branch & Master

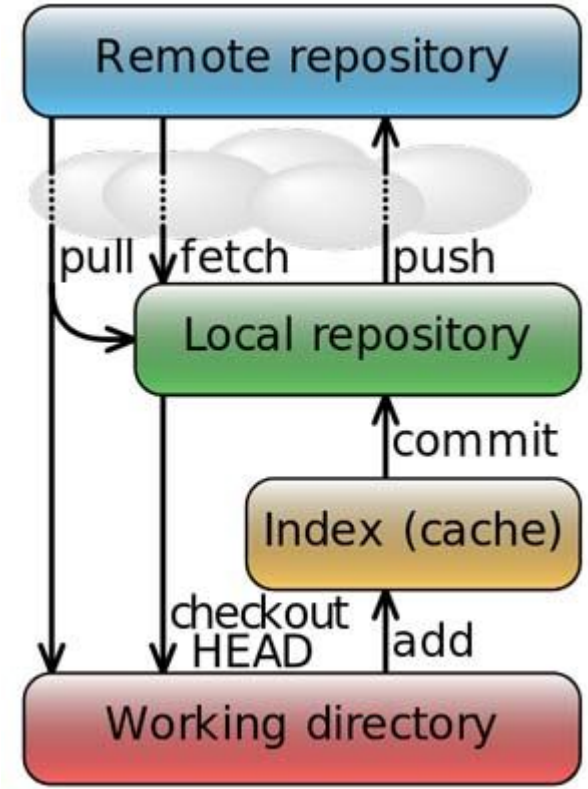
- Verzija repozitorijuma koja se razlikuje od master grane projekta
- Svaki repozitorijum sadrži master granu koja je primarna za svaki repozitorijum
- Grane se koriste kako bi se omogućio razvoj funkcionalnosti, popravka bagova, lično eksperimentisanje i sl. bez mogućnosti da se glavna grana projekta pokvari sa novonastalim izmenama
- Pre nego što neka izmena završi na master grani, potrebno je detaljno proveriti da li su izmene korektne i nemaju opasnosti po glavnu granu projekta
- Proces u kome je opisan pravilan način rada sa granama i razvijanjem novih funkcionalnosti sistema zove se **Git Feature Branch Workflow**

Checkout

- Komanda sa kojom se menja trenutna grana na kojoj se nalazimo na nivou repozitorijuma
- Pravljenje nove grane i prelazak na nju
  - **git checkout -b feature-add-logout-button-to-homepage**
- Povratak na master granu
  - **git checkout master**
- Brisanje grane
  - **git branch -d feature-add-logout-button-to-homepage**
- Lokalno napravljena grana nije dostupna ostalim korisnicima repozitorijuma dok se ne uradi push na remote repozitorijum
  - **git push origin <*branch_name*>**

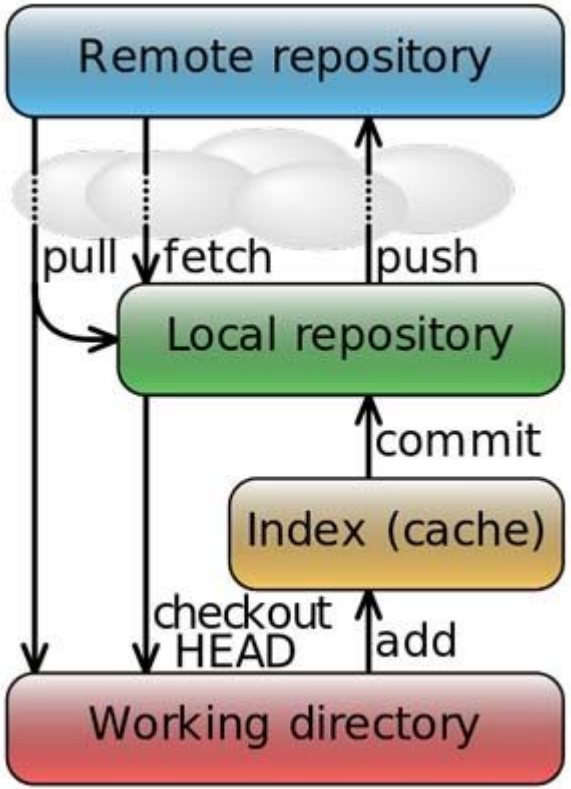
Add & Commit

- Dodavanje promena na **Index**
  - **git add <filename>**
  - **git add ***
- Komitovanje promena
  - **git commit -m *"Enter your commit message here"***
- Nakon ovoga, komitovani fajl se nalazi na **HEAD-u**
- HEAD predstavlja referencu na poslednji komit koji se nalazi na repozitorijumu. Nakon dodavanja novog komita, HEAD postaje novi komit
- Pravila prilikom komitovanja
  - Svaki komit mora da prolazi build
  - Poruka komita je jasna i precizna
  - Komiti su mali
  - Povezivanje sa ticket-om



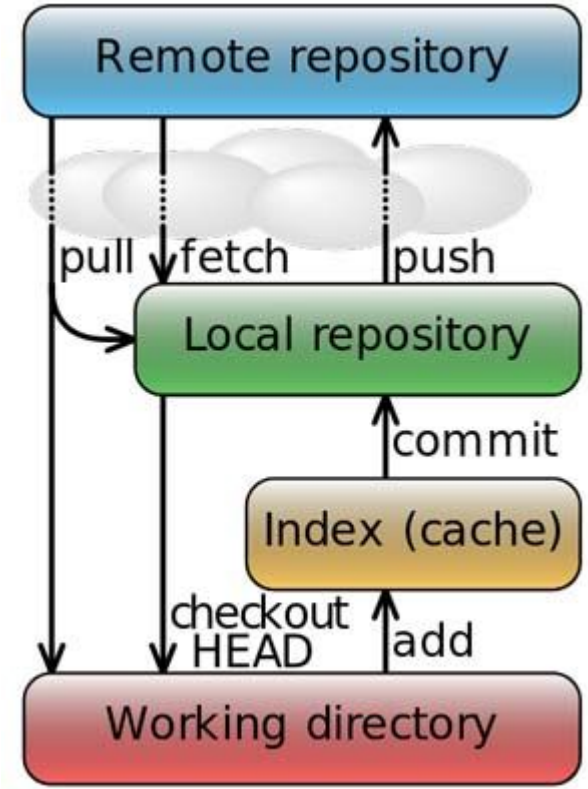
Push

- Da bi se izmene koje se trenutno nalaze na HEAD-u poslale na remote repozitorijum potrebno je
  - **git push origin <*branch_name*>**



Pull & Merge

- Ažuriranje lokalnog repozitorijuma sa komitima koji se nalaze na remote repozitorijumu
  - **git pull**
- Spajanje neke druge grane u trenutnu granu
  - **git merge <*branch_name*>**
- Prilikom spajanja grana moguće je doći do konflikata koje je potrebno razrešiti



Tag

- Dodaju se na komite
- Preporuka je da se dodaju kako bi oznacili release-ove sistema. Ovaj koncept postoji i u SVN-u.
- **git tag 1.0.0 *<commit_id>***

Log

- Predstavlja prikaz istorije repozitorijuma
- Komiti određenog autora
  - **git log --author=Pera**
- Fajlovi koji su se promenili
  - **git log --name-status**
- Za ostale parametre
  - **git log --help**

Stage, unstage, staging area

- Staging area (index)
  - Kontejner u koji git skuplja sve promene koji su deo sledećeg komita
- Staging
  - Stavljanje fajla u index-a
- Ovo je korisno kada smo menjali više fajlova a želimo da napravimo manje komite. Tada se u staging area stavljaju pojedini fajlovi (ili čak njihovi pojedini delovi ili samo jedna linija unutar fajla) koje želimo da imamo u sledećem komitu.
- Unstaging
  - Uklanjanje fajla iz index-a

Reset

- Primer: **- A - B - C (master).** Head pokazuje na C i index se poklapa sa C.
- Resetovanje svih lokalnih izmena
  - **git reset** ***[<mode>] [<commit>]***
  - **git reset --soft B**
    - HEAD pokazuje na B. Index i dalje ima promene od C. **git status** promene pokazuje kao **staged**
  - **git reset --mixed B**
    - HEAD pokazuje na B. Index je modifikovan da se poklapa sa B. I dalje imamo dostupne promene načinjene nad lokalnim fajlovima koje se ne nalaze u index-u. **git status** ih pokazuje kao **unstaged**.
  - **git reset --hard B**
    - HEAD pokazuje na B. Index je modifikovan da se poklapa sa B. Sve izmene nad lokalnim fajlovima se uklanjaju i fajlovi imaju isti sadržaj kao na komitu B.

.gitignore

- Fajl u kom se navode fajlovi i folderi na nivou projekta i za koje ne želimo da pratimo promene
- Za generisanje može se iskoristiti https://www.gitignore.io/

Git Feature Branch Workflow

- Ideja je da se razvoj novih funkcionalnosti kao i popravka postojećih odvija na posebnim granama umesto direktno na **master** grani.
- Ova enkapsulacija omogućava da više programera radi na istoj funkcionalnosti bez da se glavni codebase ažurira pre završetka nove funkcionalnosti.
- Korišćenje grana tera programera na upotrebu **pull request-ova**
- **Pull request** ne mora da služi samo kao krajnja tačka ovog procesa u kojoj će se uraditi code review i functional acceptance. Nasuprot ovome, pull request se može otvoriti pre kraja implementacije kako bi se diskutovalo o trenutnom stanju sa kolegama i dobila neka sugestija, pomoć, itd.
- Za potrebe fakultetskog projekta biće nam dovoljna upotreba **feature** i **bugfix** grana

Git Feature Branch Workflow

1. Prebacivanje na **master** granu
   a. **git checkout master**
   b. **git fetch origin**
   c. **git reset --hard origin/master**
2. Pravljenje željene feature ili bugfix grane
   a. **git checkout -b *feature-add-logout-button-to-homepage***
   b. **git checkout -b *bugfix-add-logout-button-to-homepage***
3. Rad na novoj grani
   a. Add, commit, push, pull, itd.
4. Otvaranje pull request-a
   a. Code review
   b. Functional acceptance
5. Merge na **master** granu

Git GUI

- Postoje alati koji nude grafički prikaz svih prethodno pomenutih funkcionalnosti Git-a. Neki od njih su
  - SourceTree
  - GitHub Desktop
  - TortoiseGit
  - ...
- Pored ovih alata mnogi IDE podržavaju integraciju sa Git-om i nude vizuelni prikaz funkcionalnosti Git-a
  - U IntelliJ IDEA postoji Version Control Tool Window

Coding conventions

- Poštovanje konvencija prilikom pisanja programskom koda omogućava da prilikom rada više programera na jednom codebase-u kod koji iskodiraju na kraju bude što je moguće više konzistentan
- Na nivou projekta moguće je redefinisati standardne konvencije zbog boljeg uklapanja u projekat