Sistemi kontrole verzija
(Version Control Systems)

Principi softverskog inženjerstva (SI3PSI, MS1PSI)

Laboratorijska vežba br. 1

Predavač: Dražen Drašković, asistent

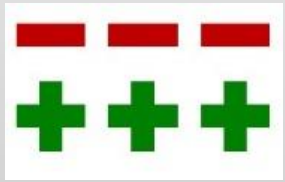
Autori: Dražen Drašković, Nenad Vitorović

Kako trenutno radimo

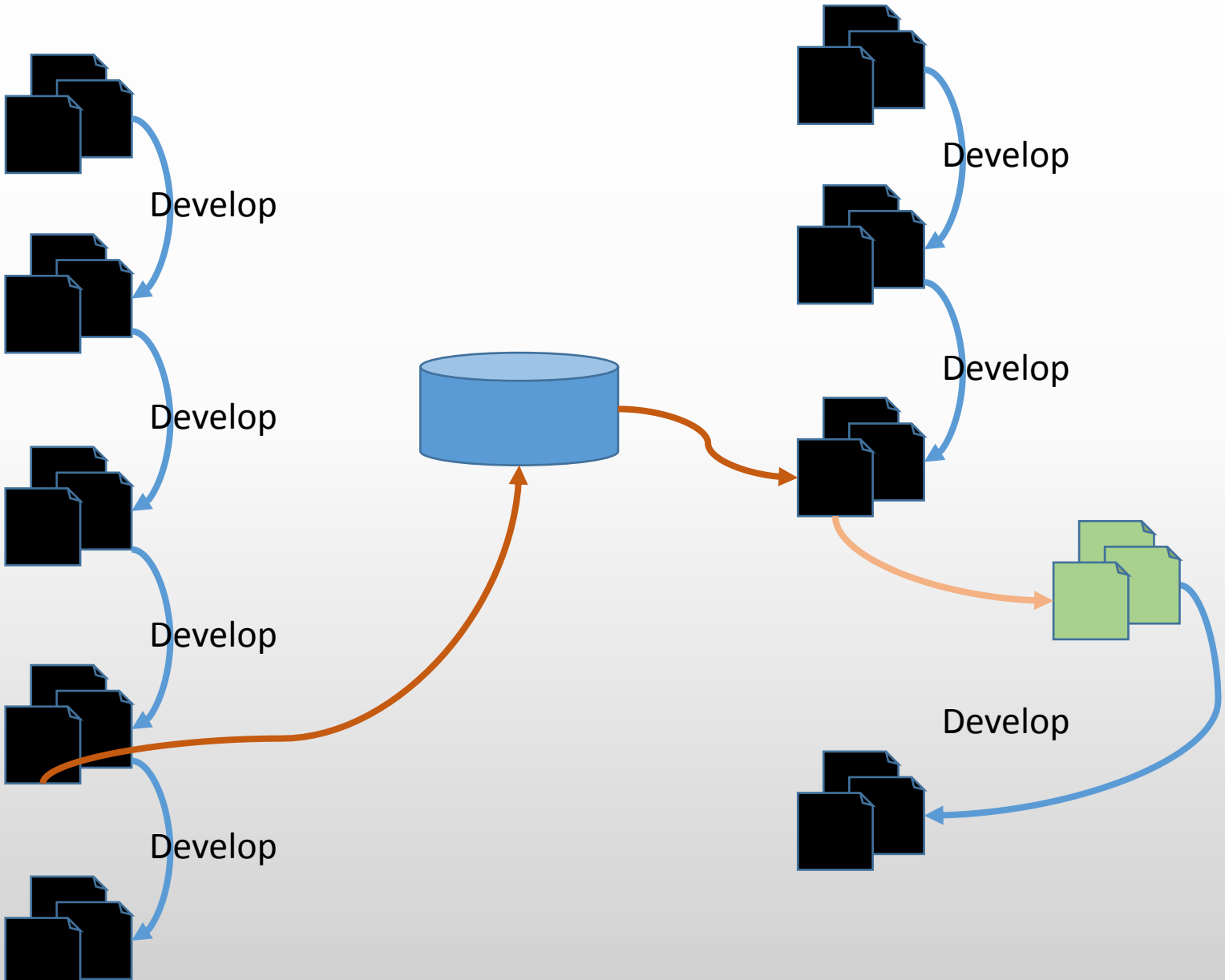
- Radimo na kodu
- Imamo smislenu celinu
  - Nešto je profunkcionisalo
  - Ispravili smo neku grešku
- Spremni smo da napravimo "snimak" toga
  - Zato što je potpuno dobro
  - Zato što nastavljamo rad, plašimo se da nešto ne "pokvarimo" (backup)
- Ako radimo u timu
  - Možda želimo da podelimo sa drugima
  - Možda još neko vreme da zadržimo samo za sebe – ne ugroziti druge!
  - Da li šaljemo kod mailom? :-/

Kako je to podržano alatima

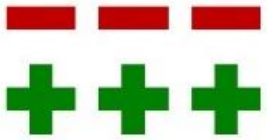
- SVN
  - Centralizovani sistem verzionisanja (VCS)
  - Podržava vođenje istorije, i pravljenje "snimaka"
  - Ne dozvoljava da u jednom momentu stanje projekta perzistiramo, a da tek naknadno podelimo to sa drugima
    - Istog momenta kada perzistiramo, to je vidljivo i drugima
- Git/Mercurial
  - Distribuirani sistem verzionisanja (DVCS)
  - Podržava isto što i SVN
  - Moguće perzistiranje stanja projekta više puta, pre deljenja sa drugima



Šta je prirodno?



GIT koncepti



- Pored centralnog repozitorijuma,
  i svaki korisnik ima **sve** kod sebe:
  - Čitavu ***istoriju izmena*** – svoje i tuđe
  - Kompletne podake o repozitorijumu
  - Nije neophodno da korisnik bude na mreži da bi koristio git (što za SVN ne važi!)
- Korisnici mogu međusobno sarađivati i mimo centralnog repozitorijuma:
  Perin radni direktorijum je udaljeni repozitorijum za Miku
- ***Working tree***: Svaki korisnik ima i svoj
  radni direktorijum: tekuće stanje,
  koje nije verzionisano (sačuvano u istoriji)

Git init

- Repozitorijum se mora INICIJALIZOVATI
  - Komanda

    `git init`

    se izvršava u direktorijumu koji želimo da služi za verzionisanje
    - Unutar tog direktorijuma, napravi **.git** direktorijum, koji sadrži sve informacije o repozitorijumu.
  - Poziva se uz opciju **bare** za pravljenje foldera pod kontrolom gita na serveru: **`git --bare init`**
    - Samo obezbeđuje da se ne vide fajlovi

Dodavanje udaljenog repozitorijuma

- Dodavanje glavnog deljenog repozitorijuma (*remote*):

```
git remote add origin URI_DO_REPO-a
```

- Remote sa nazivom **`origin`** je „glavni“ udaljeni repozitorijum.

```
git remote add origin
xy110xxxd@rtidev5.etf.rs:gitdir
```

- Moguće je dodati i druge udaljene repozitorijume, sa proizvoljnim nazivima

```
git remote add pomocni file://Dropbox/deljeni.git
```

Radni direktorijum

- Provera stanja:

`git status`

  - Da li ima fajlova koji se ne prate (nisu registrovani)
  - Da li ima promena u fajlovima koji se prate

- Dodavanje fajlova u praćenje:

`git add`

  - Moguće dodavanje pojedinačnih fajlova (navesti ime)
  - Moguće i sve dodati: **`git add .`**
  - **`git add <path_to_file>`**
    - **`Dodaje novi fajl u verzionisanje`**
    - **`Za postojeći, izmenjeni fajl,`**

Komitovanje

- ***Komitovanje:***

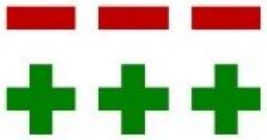
  beleženje tekućeg stanja jednog ili grupe fajlova u „istoriji“

- ***Komit (commit):***

  Jedna tačka u istoriji git repozitorijuma, koja sadrži skup jednog ili grupe fajlova (predstavlja verziju)

- ***Stage, Index***: **podskup** izmena koje su napravljene u odnosu na prethodni commit, a koje su **odabrane** za novi commit
  - Promene “odabrane” preko **`git add`**
    - To su novi fajlovi i izmene postojećih
  - Ne nužno SVE načinjene promene – moguća je selekcija navođenjem putanja do željenih fajlova (pravi podskup)
- Prilikom izvršavanja operacije commit, samo promene koje su na stage-u ulaze u istoriju, ostale promene ne

Komitovanje



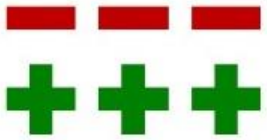
`git commit`

- Celina koju bismo sačuvali je u radnom direktorijumu
  - Pre **`commit`**, izmenjeni fajlovi moraju se *stage*-ovati, komandom **`git add`**, koja se ovog puta odnosi na sadržaj izmena fajla
- Fajlovi koji nisu staged, ne bivaju komitovani
- Moguće odabrati samo deo fajlova čije stanje želimo da čuvamo
- Neophodno ostaviti poruku –sam git ne dozvoljava drugačije
  - Argument:

```
git commit  -m "Tekst poruke za komit"
```

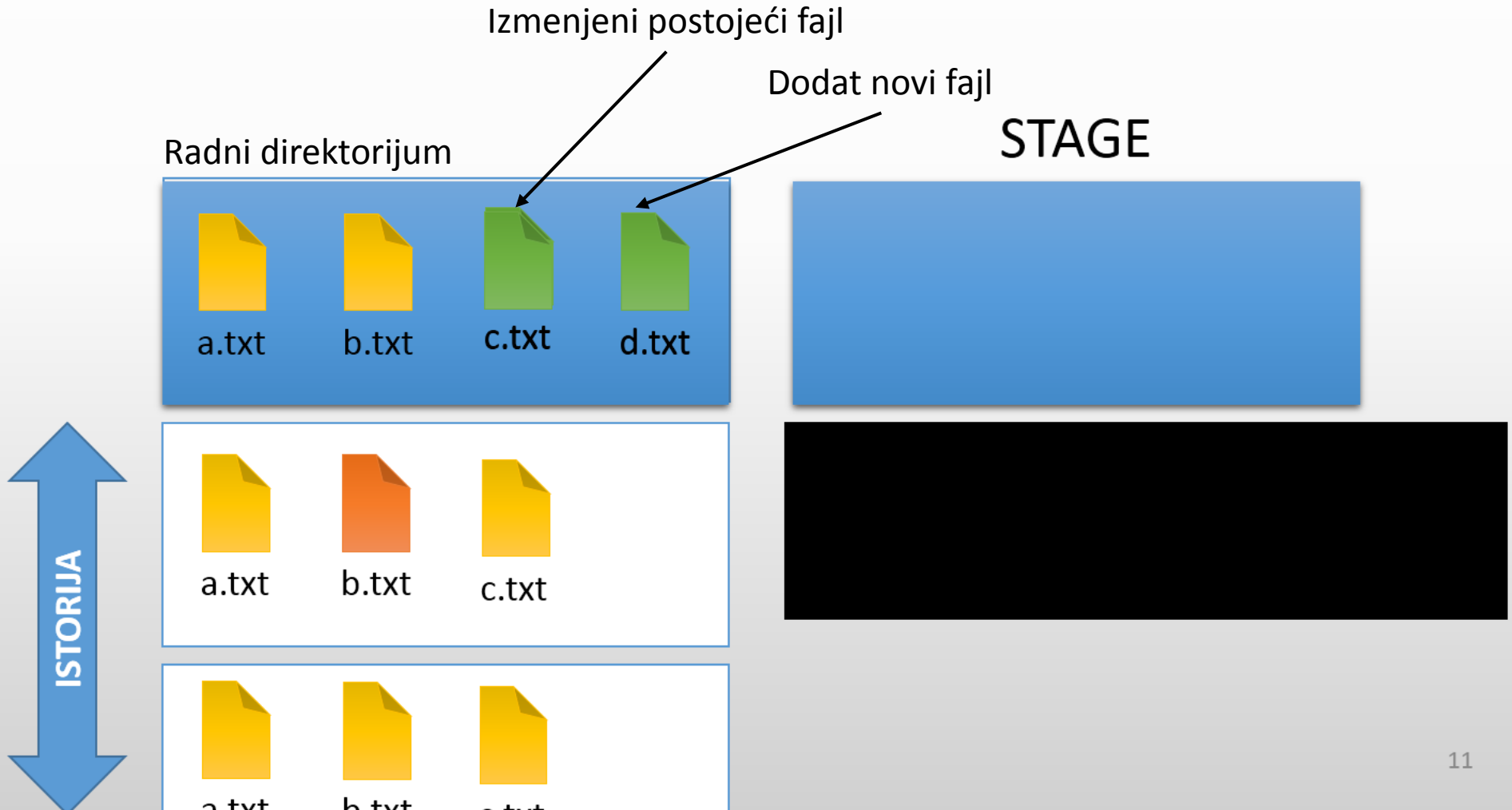
- Preskakanje stage-a: **`git commit -a -m "..."`**

Praktičan primer



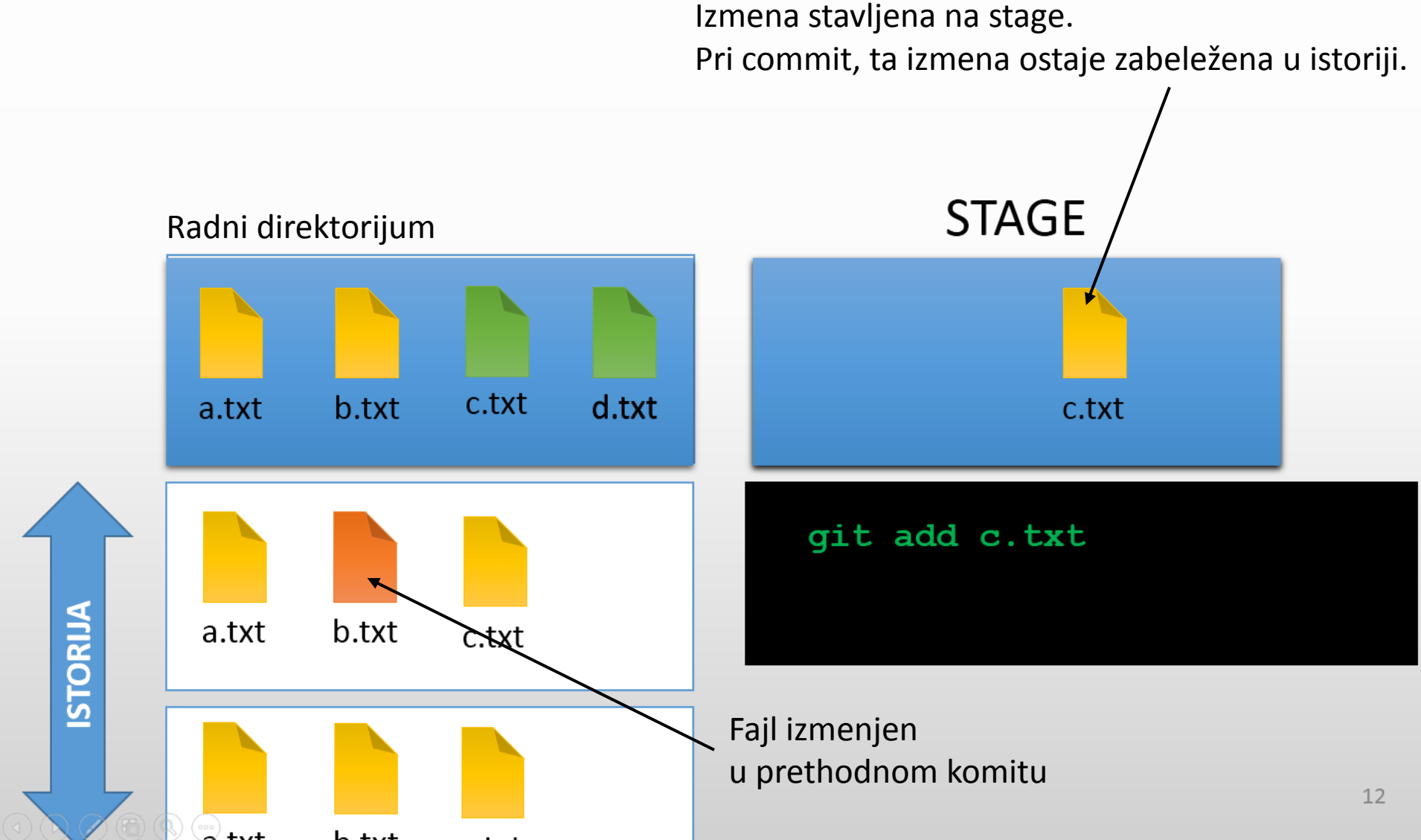
- Napravimo index.html
- Izmenimo nekakav tekst
- **`git status`**
- **`git add index.html`**
- **`git commit –m "Prvi commit"`**
- Napravimo contact.html
- Izmenimo index.html
- **`git status`**
- **`git commit –m "Drugi commit"`**
- Šta je na repozitorijumu? Šta drugi vide?
- **`git log`**
- **`git status`**

Stage i working tree – „ispod haube“

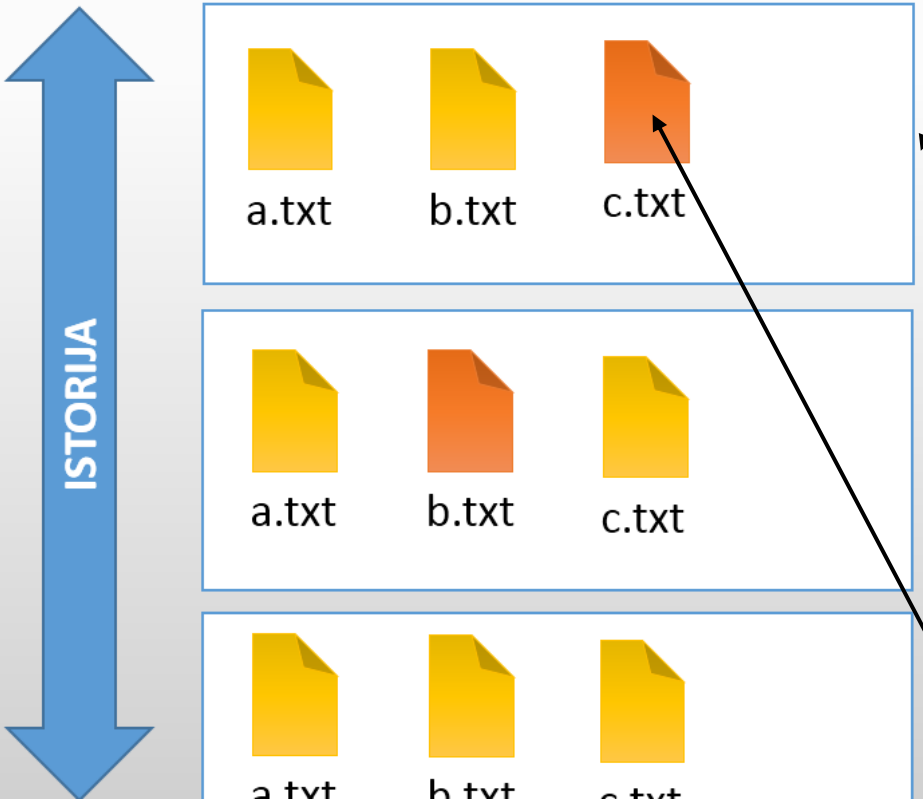


Stage i working tree – „ispod haube“

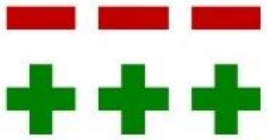
Izmena stavljena na stage.
Pri commit, ta izmena ostaje zabeležena u istoriji.



Stage i working tree – „ispod haube“



Kako radi git?



- Prati promene u fajlovima
- Za svaki fajl koji želimo commit-ovati
  - Na staging-u, git locira promene u fajlovima: koji red je izbačen, koji red je ubačen
  - Jedan commit predstavlja promene koje smo načinili nad poslednjim commit-om
  - Efektivno, evidentiraju se *razlike* u fajlovima
  - Razlike u istoriji: **`git log -p`**
  - Razlika –*diff*

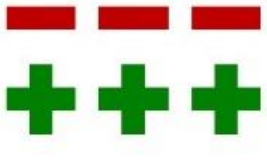
```
index 17ec217..f056f03 100644
@@ -117,27 +117,24 @@ class OfferController {
         fixupBasicDetailsCommand(basicDetailsCommand)
         if (basicDetailsCommand.validate()) {
             Offer.withTransaction {
-                BasicDetails bd = new BasicDetails()
-
-                bindData(bd, basicDetailsCommand)
-                log.error("Reward: $bd.condition $bd.reward")
+                BasicDetails bd = new BasicDetails(basicDetailsComm

+                log.debug("Condition: ${bd.condition?.value}:${bd.c
                 Offer offer = new Offer(basicDetails: bd)

                 offer.advertiser = collaboratorService.currentAdver
                 offer.save(failOnError: true)
+                log.debug "Offer saved successfully"
             }
         } else {
-            log.debug "${basicDetailsCommand.errors}"
```

Uklanjanje i preimenovanje fajlova

- Uklanjanje: **`git rm filename`**
  - Ručno uklonjen fajl – ipak izvršiti ovu komandu (čak iako više ne postoji), kako bi ga git uklonio iz evidenicije
- Zaustavljanje praćenja, uz zadržavanje fajla u lokalu:

  `git rm --cached filename`
- Preimenovanje: **`git mv filename`**
  - Ili ručno preimenovanje, a zatim
    - **`git rm oldName`**
    - **`git add newFile`**

Poništavanje promena



- Skidanje fajla sa stage-a:

```
git reset HEAD <file>
```

  - Fajl ostaje izmenjen, samo ne učestvuje u komitu
  - *Undo add*, efektino
- Poništavanje promena

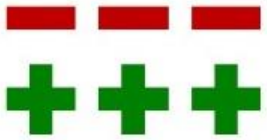
```
git checkout -- <file>
```

  - Fajl se vraća na stanje u kom je bio u prethodnom komitu (sve izmene se anuliraju)
  - Moguće specificirati sa kog od ranijih komitova se uzima fajl
- Vraćanje unazad, na neku od ranijiih revizija (vraćanje kroz vreme):

```
git checkout 0d1d7fc32
```

Heksadecimalna oznaka revizije, dostupna preko git log

Rad sa drugima



- Potreban "*centralni*" repozitorijum, deljen sa drugima
- Inicijalno dovlačenje sa udaljenog (deljenog) repo-a:

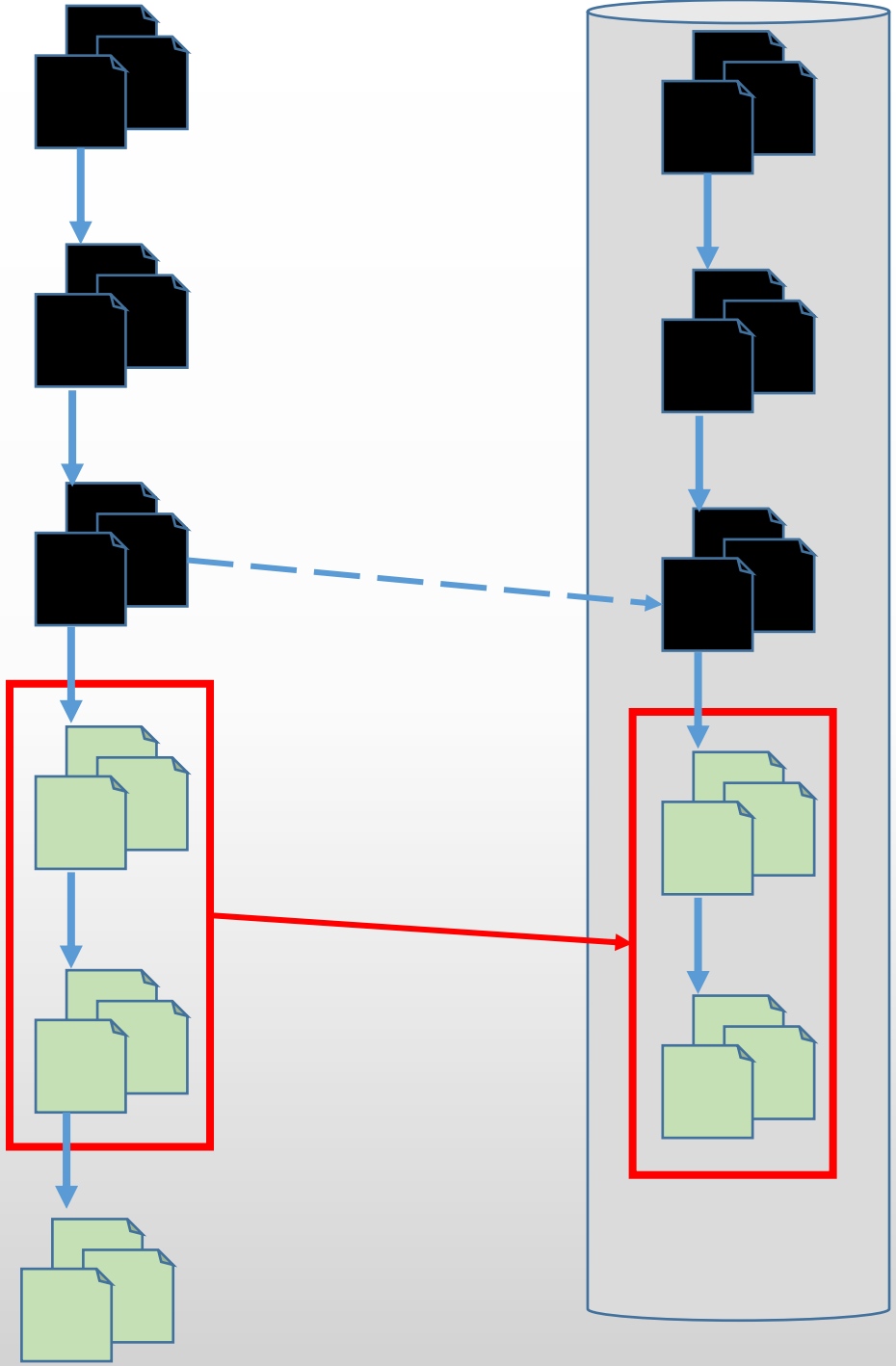
```
git clone <path_to_repo>
git clone short_remote_name
```

- Kreira direktorijum istog naziva kao udaljeni repozitorijum,
  - unutar njega se nalazi **.git** direktorijum
  - i svi fajlovi koji pripadaju tom repozitorijumu.
- Dovlačenje promena: **`git pull`**
  - Dovlači promene sa udaljenog repozitorijuma, i spaja ih sa komitovima koji nisu poslati na taj repozitorijum
- Odakle se dovlači?
  Podrazumevano, radi se sa repozitorijumom koji smo klonirali (origin), ako se ne zada eksplicitno putanja ili naziv remote repozotorijuma

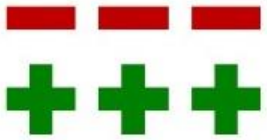
Rad sa drugima

- Slanje promena: **`git push`**
  - Šalje na udaljeni repozitorijum sve načinjene komitove, koji nisu na njemu
  - Moguće poslati promene na neki od udaljenih repozitorijuma
- Gde se šalje? Podrazumevano, radi se sa repozitorijumom koji smo klonirali (origin), ako se ne zada eksplicitno putanja ili naziv remote repozotorijuma

Čuvamo dalje izmene,
verzionišemo ih,
ali NE ZATRPAVAMO DRUGE
neproverenim kodom

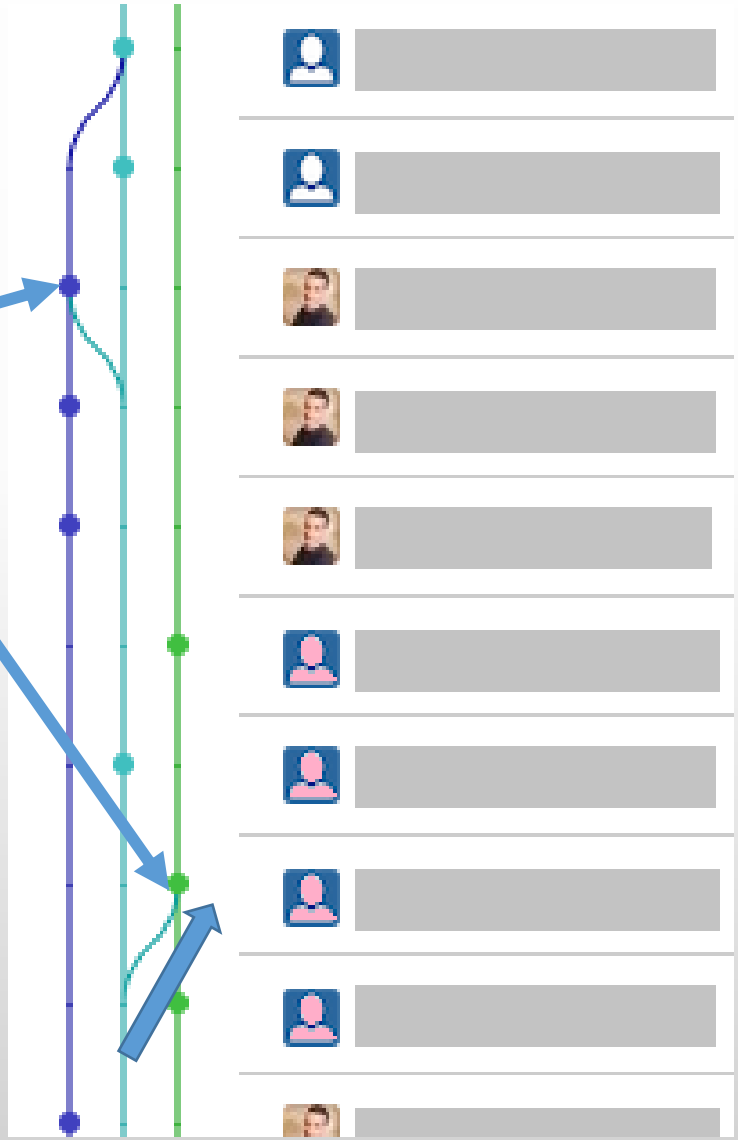


Rad sa drugima



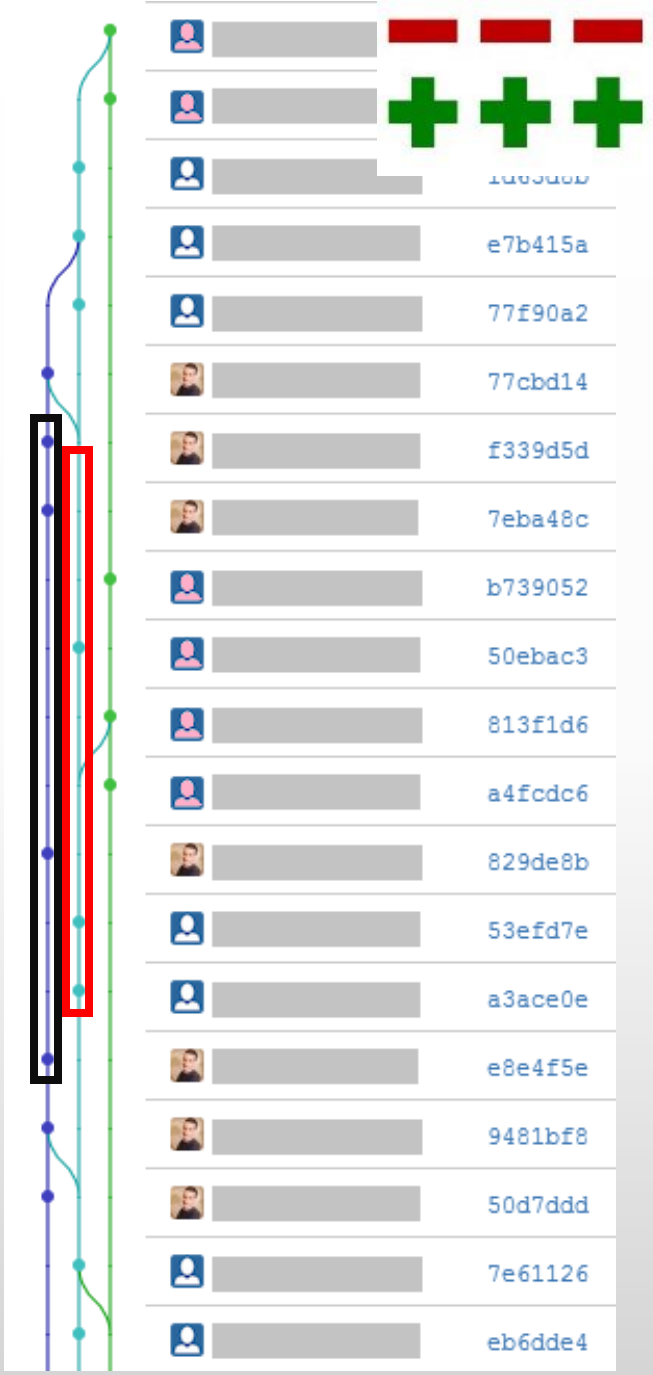
- Izmene nad istim fajlom
  - Osoba koja *pull*-uje poslednja, mora uraditi merge
  - *Merge* – spajanje dva stabla izmena: lokalno stablo sa stablom sa udaljenog repozitorijuma
  - Git pokušava automatski da uradi merge
- Ako pri *pull* ne uspe automatski merge, neophodno ručno
  - Kada se dešava neuspeh automatskog merge-a?
    - Izmenjena ista linija koda – konflikt;
      alat ne uzima odgovornost za merge => moramo ručno
  - Git prikazuje upozorenje
  - Komanda **`git mergetool`** pokreće alat za merge konflikata (Meld, Diff3,…)
- Kada se završi merge (bilo automatski, bilo ručno, razrešenjem konflikta),
  potrebno je napraviti commit za završeni merge

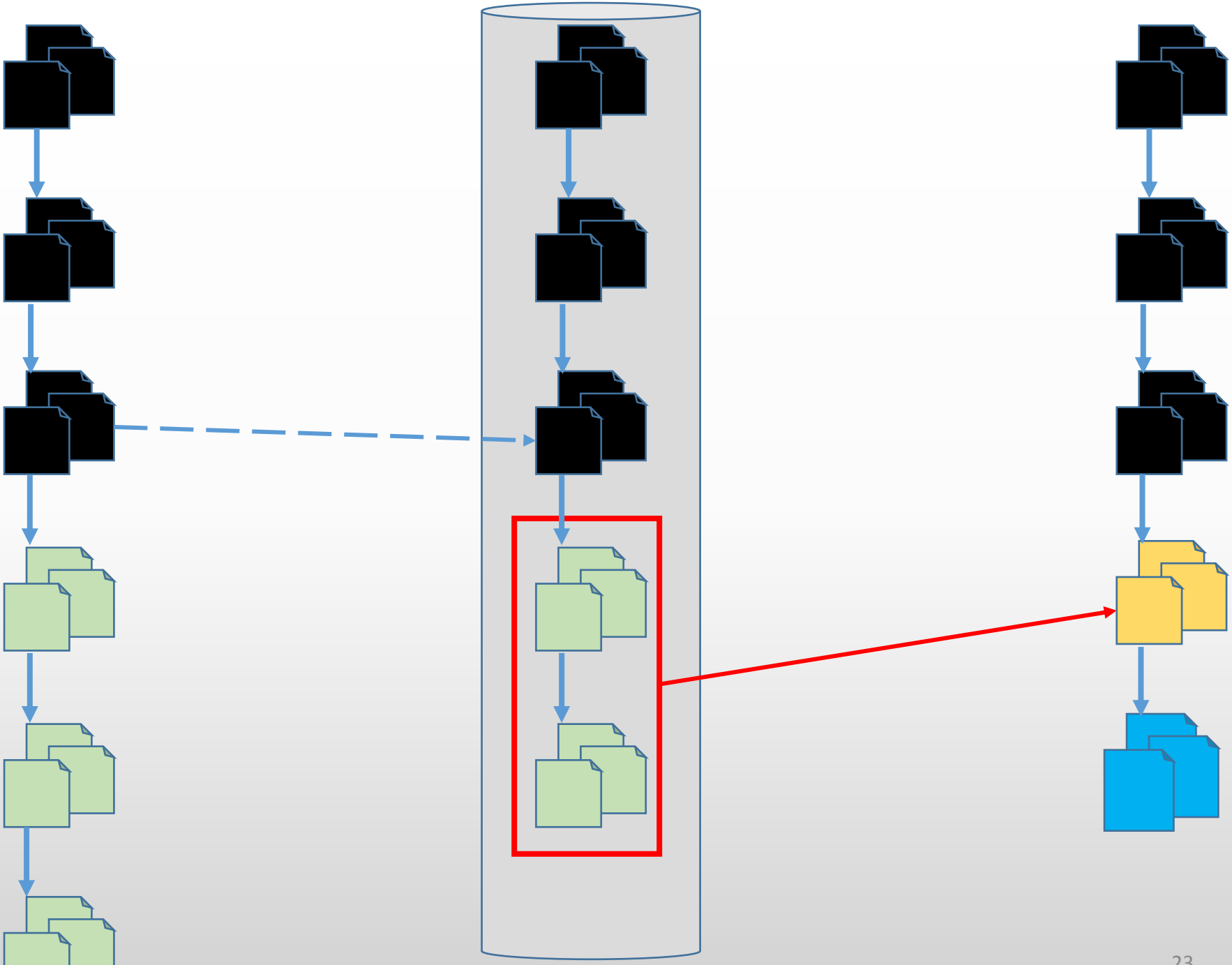
Kako to izgleda?



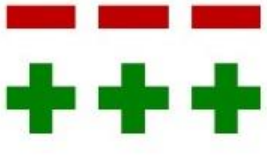
Commit nastao merge-om

Urađen „pull“ sa stabla dev 2 (svetloplava) na stablo dev 3 (zelena)





Razni git koncepti



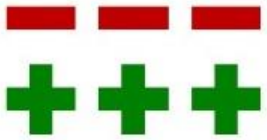
- Remotes
  - Push/pull: na repozitorijum koji je kloniran, podrazumevano
  - Pored "centralnog" repozitorijuma, moguće koristiti i druge repozitorijume (istog projekta).
    - Npr, kolega koji je klonirao centralni repozitorijum želi da podeli nešto samo sa vama => koristiti njegov repozitorijum za pull

```
git remote add short_remote_name remote_uri
git push short_repo_name
```

- Tagging: oznavačanje verzija koje predstavljaju koherentne celine (milestones npr.).
  - Mogući proizvoljni nazivi. Uglavnom se koriste nomenklature verzionisanja SW.
- Stash
  - Pull i merge uspevaju jedino sa čistim radnim direktorijumom
  - Ili sve komitovano, ili gurnuto u stranu (stashed)
- Ignorisanje: napraviti `.gitignore` fajl u korenu git repozitorijuma; sadrži putanje do fajlova koje ne želimo da nikada add-ujemo i pratimo ili izraz koji to čini za grupu fajlova

```
*.png #nijedna slika
application/log/* #nijedan fajl iz log dir
application/random/todolist.txt #konkretni f
```

Branching



- Paralelni razvoj u više nezavisnih tokova unutar istog repozitorijuma - ***branching***
- Uvek postoji glavna razvojna grana (**`main`**)
- Postoje specijalno kreirane grane,
  obično za nestabilne izmene, ili radi nezavisnog razvoja različitih funkcionalnosti

```
git checkout <branch_name>
```

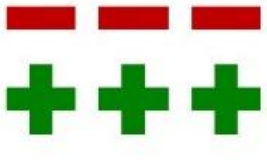
- Naredba **`checkout`** kreira granu sa nazivom **`branch_name`**, ako ona ne postoji,
  odnosno radni direktorijum „prelazi“ u tu granu, ukoliko već postoji.

Branching

- Dalji commit-ovi predstavljaju istoriju te grane, nezavisnu od istorije main grane
- Povratak na **main** granu: **`git checkout main`**
- Grane se mogu **push**-ovati i **pull**-ovati, kako bi više ljudi moglo da radi na različitim granama.

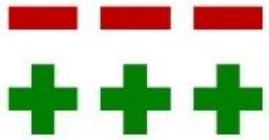
- Kada je paralelna grana „zrela“, ona se spaja sa glavnom razvojnom:
  - pozicioniramo se na **main** granu, uradimo **pull** sa odvojene grane
  - Spajanje dve grane gotovo identično kao **pull** sa udaljenog repozitorijuma!

Git tok rada –rekapitulacija



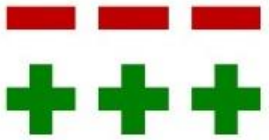
1. **`git clone repo`**
2. **`git add <untracked_file>`**
3. **`git add <changed_file>`**
4. **`git commit –m "..."`**
5. Ponavljati 2..4 više puta
6. **`git pull`**
   1. **`git mergetool`**, ako postoje konflikti
   2. **`git commit`**, onoga što se dobilo spajanjem lokalnih sa udaljenim izmenama
7. **`git push`**, opciono, nekada, kada je potrebno podeliti izmene sa drugima.

Napomene



- Samo fajlovi se registruju, ne i direktorijumi
- Ne postoji način da se verzioniše direktorijum, samo fajlovi u njemu
- Alati:
  - #1 - Source Tree, za Mac/Windows
  - GitCola, zahteva python, Za Linux
  - Unutar IDE:
    - phpStorm (Jetbrains) –odlična integrisana podrška
    - podrška u Eclipse-derived alatima kroz egit plugin
  - TortoiseGit –relativno nepregledno
  - Terminal/konzola
    - Daleko ugodnije na Linux-u

Prednosti git-a



- Bolja podrška paralelnom razvoju
- Bolja izolacija
  - Moguće eksperimentisati, praviti komitove a da se ništa ne „poruši“ drugima pritom
- Nije neophodna mreža
- Branch: jako efikasno, bez previše „mučenja“ korisnika
- Daleko brži od SVN
  - Sve na lokalnoj mašini, pa je pregled istorije mnogo brži, kretanje kroz revizije je daleko brže...

SVN

- Klijent-server arhitektura
- Server pamti sve promene ikada napravljene nad datotekama
- Podržava pomeranje i kopiranje fajlova
- Moguće praćenje praznog direktorijuma
- Omogućuje tagging
  - Verzija koja se smatra stabilnom se "označi": 1.0.0, ...
- Moguć branching
  - Od glavnog razvojnog stabla, odvaja se nova kopija tekućeg sistema
  - U paraleli, nezavisno, razvijaju se i glavna (originalna) i nova grana

SVN koncepti

- Manipuliše direktorijumom, datotekama, i prati verzije
- Slično kao git, svaki fajl ponaosob
- Repozitorijum se ne posmatra kao celovit, već svaki direktorijum nezavisno predstavlja repozitorijum
- Svaki korisnik ima samo radni direktorijum –lokalnu kopiju, a jedino mesto gde se nalaze podaci o izmenama koje su nastale je centralni repozitorijum!
  - Neophodna mreža da bi se videla istorija izmena i sl.

Operacije

- **`svn import <repository_uri>`**
  - Kreira ceo radni direktorijum kao jednu verziju koda u repozitorijum (ova komanda se koristi kada počinjete projekat). Sadržaj radnog direktorijuma je najnovija verzija na repozitorijumu.
- **`svn checkout`**
  - Kopira najnoviju verziju iz repozitorijuma u radni direktorijum
- **`svn add`**
  - Dodaje novi fajl koji ranije nije bio verzionisan.
  - Naći će se u repozitorijumu pri narednoj operaciji commit.
- **`svn update`**
  - Ažurira radni direktorijum najnovijom verzijom iz repozitorijuma (može se navesti i tačno kojom verzijom – kretanje kroz istoriju).
  - Moguće je da dođe do konflikta pri spajanju (slično kao kod git).
    Svi konflikti se moraju razrešiti pomoću merge alata.

Operacije

- **`svn commit`**
  - slanje radne verzije u repozitorijum, tako da ona postane nova glavna tj. tekuća revizija fajla/direktorijuma.
  - Moguće je komitovati određeni poddirektorijum ili određene datoteke, njihovim navođenjem u nastavku komande
- **`svn status`**
  - Prikazuje razike radnog direktorijuma sa stanjem na repozitorijumu
- **`svn remove <file>`**
  - briše fajl i obeležava ga za brisanje (ovaj fajl će biti obrisan i iz repozitorijuma pri sledećoj operaciji commit)
- **`svn cp <file>`**
- **`svn mv <file>`**

Minimizovanje konflikta pri spajanju (merge)

- Pri izvršavanju operacije update, moguće je da dođe do konflikta.
- Konflikt nastaje ako je na repozitorijumu već izmenjen fajl koji je izmenjen i u lokalnom radnom direktorijumu.
  - Organizovati podelu posla tako da se izbegne da više članova tima radi nad istim fajlom.
- Koliko često raditi commit sa SVN?
  - Previše često nije dobro - mnogo grešaka u radu;
  - Retka upotreba nije dobra - rad sa neažurnim kodom, gubljenje vremena na rešavanju problema koji su drugi članovi tima već otklonili;
  - Ne postoji čvrst konsenzus, jedino dobro pravilo za upotrebu u praksi je da pre zadavanja operacije commit budemo sigurni da se kod koji upisujemo u repozitorijum može kompajlirati.
  - U slučaju git, operacija commit nema ovih problema, jer nije istog trenutka vidljiva ostalim članovima tima, a daje novu tačku u istoriji projekta.

SVN: Tok rada

- Pre unošenja izmena u radnu verziju koda, prvo treba uraditi UPDATE, da ne bismo menjali neažurnu verziju koda, što opet može da dovede do velikih konflikata pri spajanju.
  - Kod git, ovo nije slučaj

1. **`svn checkout <REPO_URI>`**
2. `# izmene fajlova`
3. **`svn commit`**
4. `… #novi radni dan`
5. **`svn update`**
6. `# izmene fajlova`
7. **`svn commit`**
   1. Ako postoji novija verzija, **`commit`** neće uspeti
      1. **`svn update`** #dovući najnoviju verziju sa repozitorijuma
      2. Merge (spajanje, ručno prihvatanje promena)
      3. **`svn commit`**
8. **`if(!kraj radnog dana) goto 6`**
9. `# kraj radnog dana`
10. **`goto 4`**

Šta koristiti?

- Servisi:
  - **Github** - social coding, jedino git, javni repozitorijumi, provatni se plaćaju
  - **BitBucket** - git, mercurial: mogući free javni i privatni, do 5 osoba u timu – free
  - **Google Code - gasi se u avgustu 2015. godine!!!**
    (preporuka: migrirati na GitHub, Bitbucket, i SourceForge, ručno ili uz automatske alate)
- GIT, SVN i Mercurial mogu raditi i sa fajlsistemom, samo koristiti folder koji se deli nekim od servisa kao centralni repo
  - Obezbediti da vam neko pristupa repozitorijumu: SSH
  - Dropbox, Google drive (moguće, probajte), box…