Revision control

Kontrola revizije

- **Kontrola revizije** (eng. ***Revision control (RC)***, ***Version control (VC)***) je proces upravljanja, praćenja i kontrole izmena i verzija podataka (dokumenata, fajlova, ...)
- Svaka izmena se naziva **revizijom** (eng. ***revision***) i identifikovana je nekim ID-jem
- U softverskom inženjeringu, kontrola revizije se najčešće vrši nad izvornim kodom
- Alati ili aplikacije pomoću kojih se upravlja verzijama koda, i pomoću kojih se izmene u kodu prate i kontrolišu su **Sistemi za kontrolu revizija** (eng. ***Revision Control Systems***, ***RCS***)

Sistemi za kontrolu revizije

- **Sistem za kontrolu revizija** je softverska implementacija **RC**-a koja omogućava i automatizuje proces nadzora, identifikovanja, čuvanja, spajanja i primenjivanja izmena u fajlovima
- Zašto koristiti RC sisteme?
  - Anotacija – beleže se korisnici koji izvršavaju izmene, svaka izmena ima svoj ID i svaka izmena može imati prateći komentar
  - Reverzibilnost – mogućnost jednostavnog vraćanja na neko prethodno stanje, tj. vraćanje na prethodne verzije pre određenih izmena
  - Konkurentnost – mogućnost višekorisničkog rada na istim fajlovima ili mogućnost da svaki korisnik radi na svojim kopijama fajlova

Terminologija RC sistema

- **Repository** (repozitorijum/repo) – lokacija ili folder gde se nalaze interni metapodaci potrebni za rad sistema, sačuvana istorija izmena fajlova, kao i fajlovi izvornog koda (opciono)
- **Working copy** – lokalna radna kopija fajlova preuzeta sa nekog repozitorijuma
- **Trunk** – glavno stablo/sekvenca sa revizijama fajlova
- **Branch** – odvojena i nezavisna sekvenca dobijena kreiranjem nove sekvence ili dobijena nakon grananja postojeće sekvence
- **Head** – poslednja revizija na repozitorijumu tj. vrh sekvence
- **Conflict** – situacija kada postoje dve različite izmene koje su izvršene od strane dva različita korisnika u istom fajlu

Operacije RC sistema (Git terminologija)

- **Clone** – kreiranje radne kopije na osnovu podataka preuzetih sa nekog (udaljenog) repozitorijuma
- **Commit** – smeštanje izmena sa radne kopije na lokalni repozitorijum
- **Fetch** – učitavanje izmena sa udaljenog repozitorijuma
- **Switch/checkout** – prebacivanje sa branch-a na branch
- **Merge** – spajanje i primenjivanje dve različite revizije koje se nalaze na dva različita *branch*-a
- **Pull** – Učitavanje izmena sa udaljenog repozitorijuma i njihova primena na radnu kopiju (Fetch + Merge)

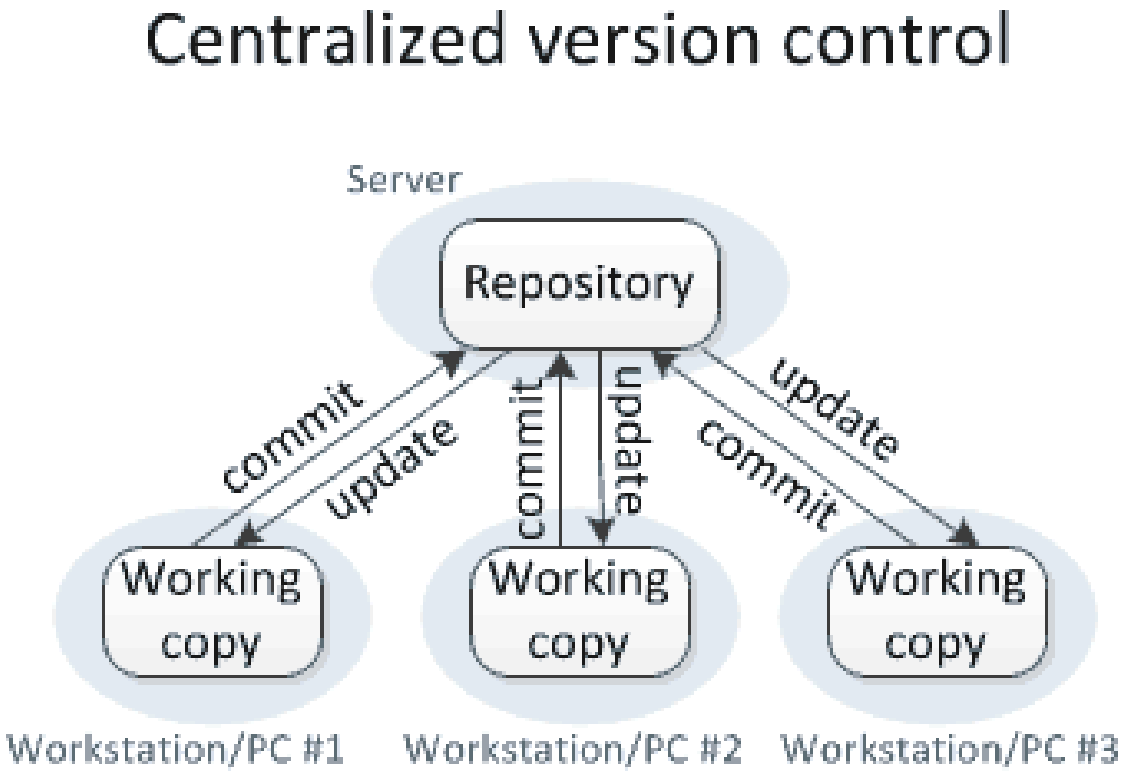
Podela RC sistema

- RC sistemi mogu se podeliti u sledeće dve grupe:
  - Centralizovane
  - Distribuirane

Centralizovani RC sistemi

- Postoji jedan referentni centralni repozitorijum na nekom serveru gde se čuvaju svi potrebni fajlovi, zajednički za sve korisnike sistema
- Na centralnom repozitorijumu se čuva istorija izmena svih fajlova
- Koriste sledeća dva mehanizma za sinhronizaciju izmena od strane više korisnika:
  - **Zaključavanje** – fajl koji se menja od strane jednog korisnika se zaključa, sprečavajući ostale korisnike da vrše izmene, sve dok korisnik ne otključa fajl koji je zaključao
  - **Spajanje** – različite izmene izvršene od strane više korisnika se spajaju i primenjuju automatski ako je moguće, ili ručno ako su izmene u istom fajlu

Centralizovani RC sistemi



Osobine centralizovanih RC sistema

- Korisnici prosleđuju svoje izmene drugim korisnicima preko centralnog repozitorijuma
- Potrebno je vršiti sinhronizaciju izmena na repozitorijumu za višekorisnički režim rada
- Kada se nove izmene preuzmu sa centralnog repozitorijuma, one se odmah primenjuju na radnu kopiju

Centralizovani RC sistemi

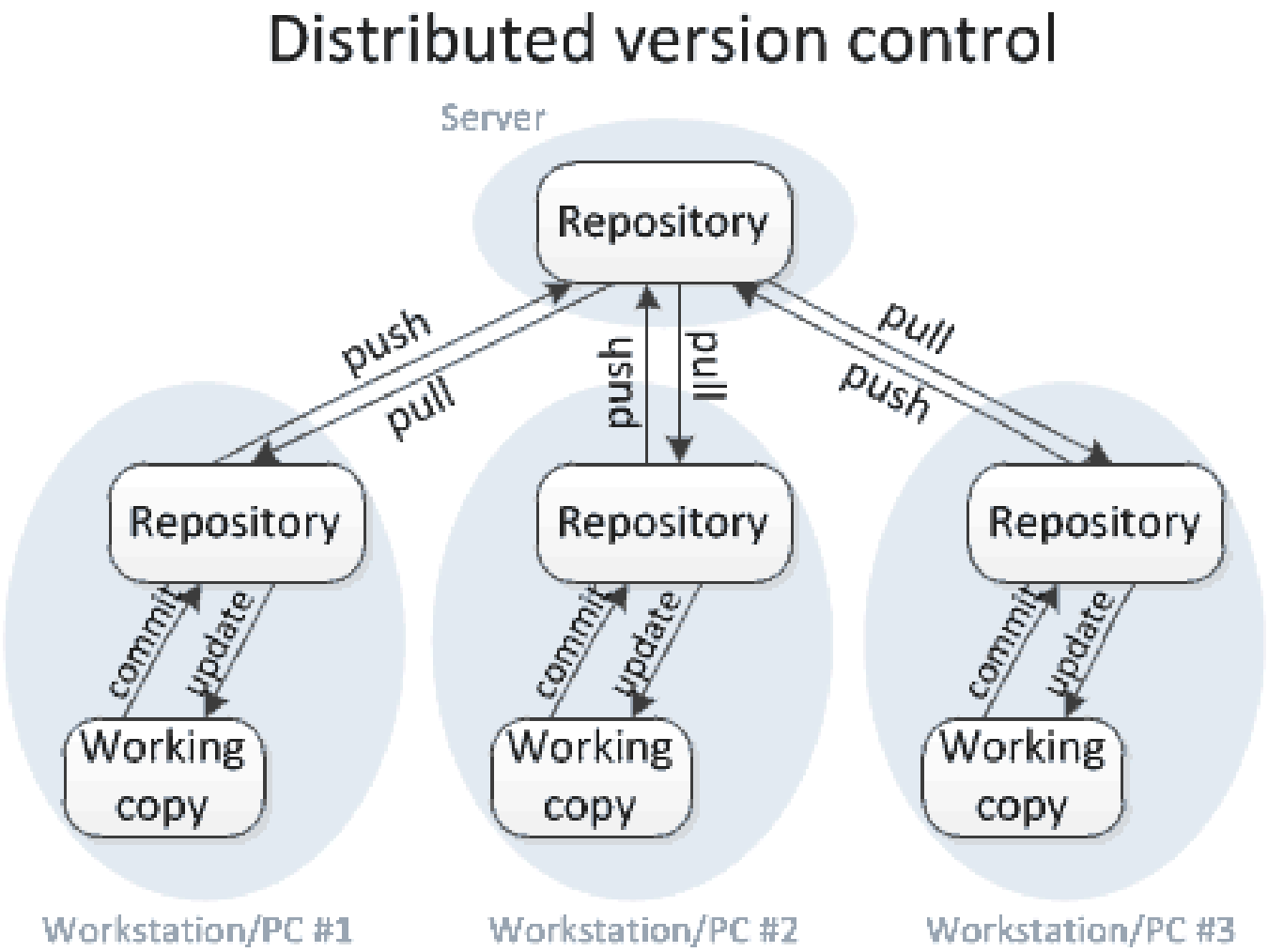
Primeri:

- Concurrent Version System (CVS)
- Subversion (SVN)

Distribuirani RC sistemi

- Svaki korisnik sistema, pored radne kopije, poseduje svoj lokalni repozitorijum
- Na lokalnom repozitorijumu su sačuvane informacije i podaci kao što je istorija izmena fajlova
- Izmene sa repozitorijuma jednog korisnika mogu se slati na/preuzeti sa repozitorijuma drugog korisnika
- Po dogovoru, jedan repozitorijum se može proglasiti referentnim za sve korisnike sistema

Distribuirani RC sistemi



Osobine distribuiranih RC sistema

- Lokalni repozitorijum se nalazi na istoj mašini gde je i radna kopija
- Komunikacija između radne kopije i lokalnog repozitorijuma se vrši *offline*, tj. ne mora se uspostaviti posebna veza sa repozitorijumom s obzirom da je na lokalnoj mašini
- Kreiranje nove revizije na lokalnom repozitorijumu i njeno objavljivanje su dve nezavisne operacije s obzirom da kreirana revizija postaje javna samo ako je preuzeta sa ili poslata na drugi repozitorijum
- Kada se izmene sa repozitorijuma jednog korisnika preuzmu i smeste na repozitorijum drugog korisnika, ne moraju se automatski primeniti na radnu kopiju drugog korisnika

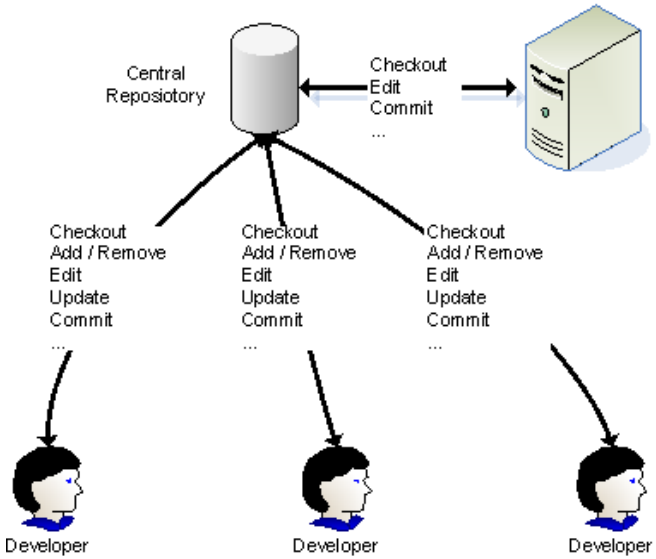
Distribuirani RC sistemi

Primeri:

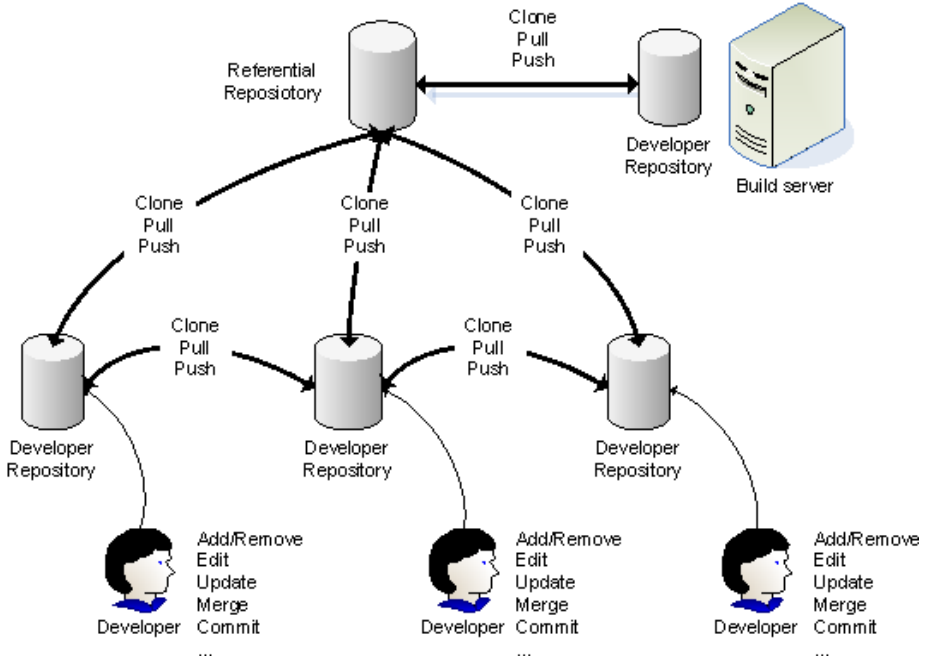
- Git
- Mercurial

Arhitektura

Centralizovani RC sistem (CVS, SVN, ...)

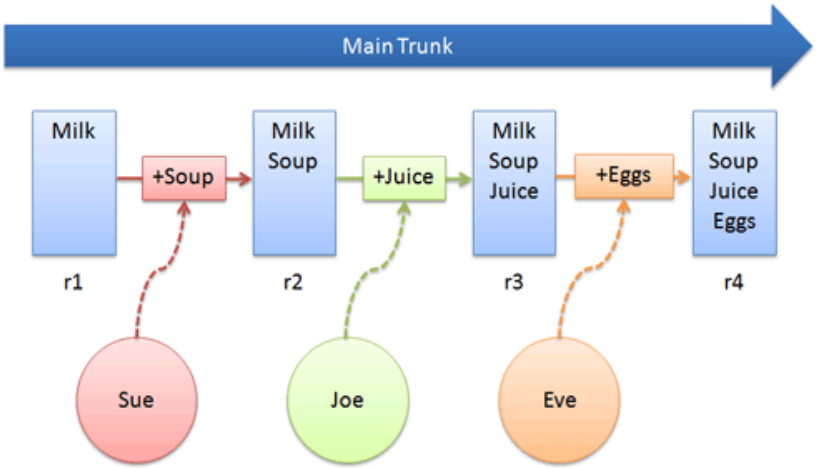


Distribuirani RC siste (Hg, Git, ...)

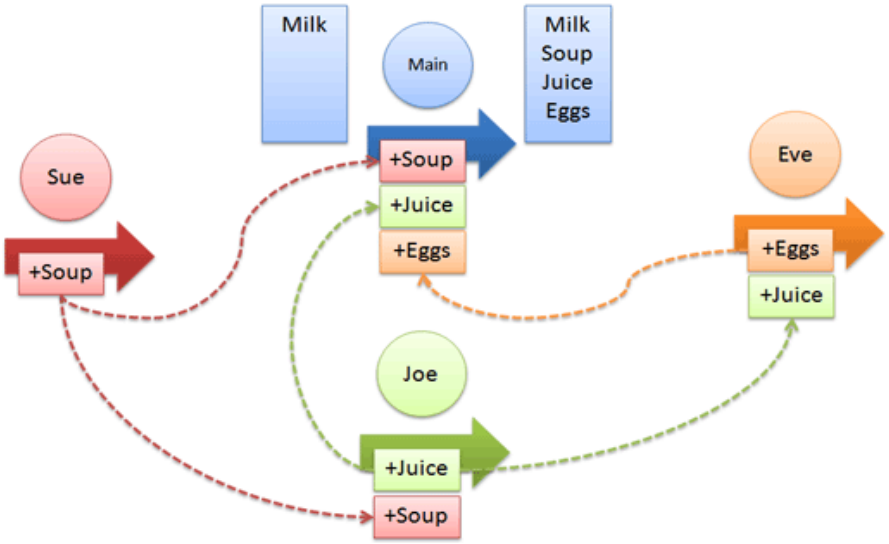


Repozitorijum

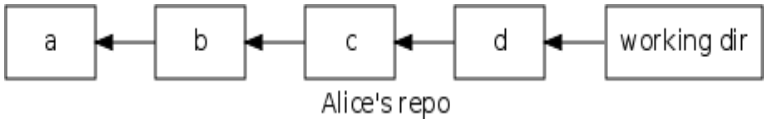
Centralizovani RC sistem



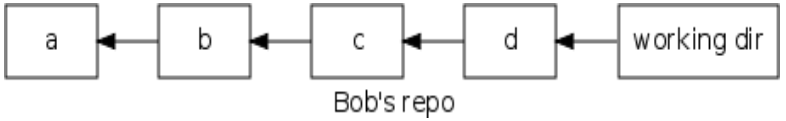
Distribuirani RC sistem

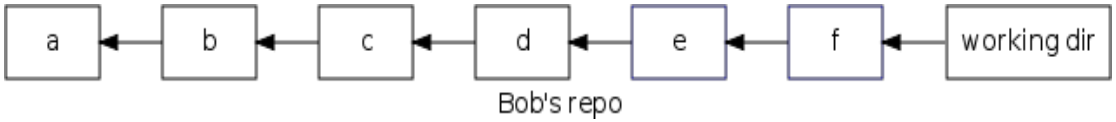


Primenjivanje i spajanje – Primer



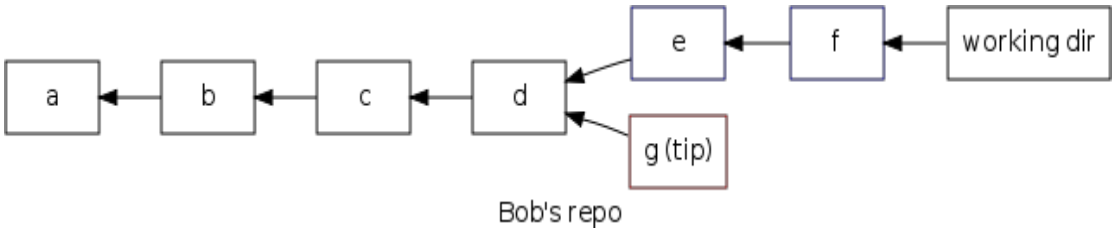
Bob klonira Alice-in repo





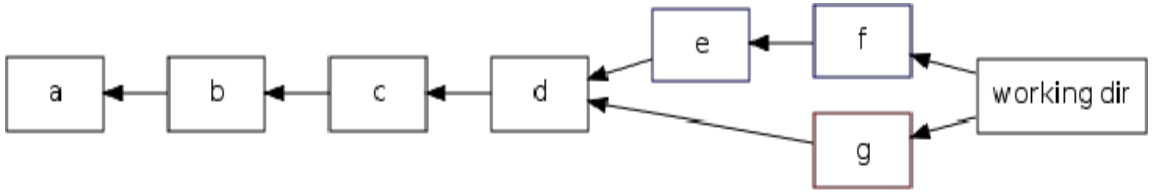
Bob izvrši izmene ***e*** i ***f*** u svojoj radnoj kopiji i commit-uje ih jedan posle drugog na svoj repozitorijum. Repozitorijum sadrži dve nove revizije, ***e*** i ***f***

Alice izvrši izmenu ***g*** u svojoj radnoj kopiji i commit-uje izmenu na svoj repozitorijum. Repozitorijum sadrži novu reviziju ***g***



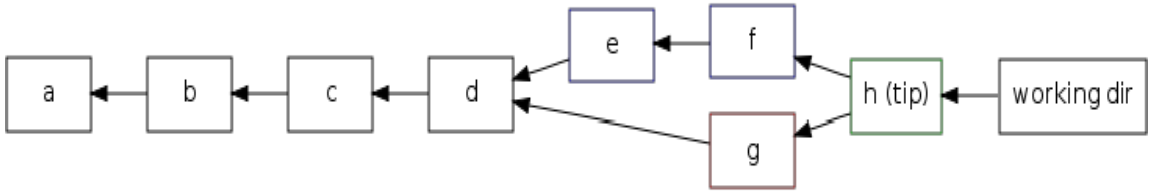
Bob fetch-uje izmene sa Alice-inog repozitorijuma da se sinhronizuju. Bob-ova radna kopija se još uvek nije promenila, tj. izmene još nisu primenjene na fajlove

Primenjivanje i spajanje – Primer (nastavak)



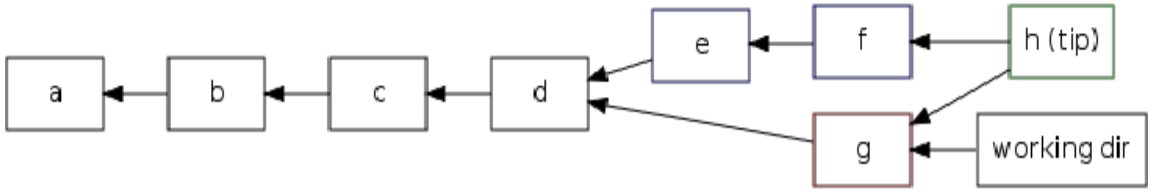
Bob's repo

Zbog toga što je Bob commit ovao svoje izmene nakon clone-a, mora da merge-uje svoje izmene sa Alice-inim izmenama



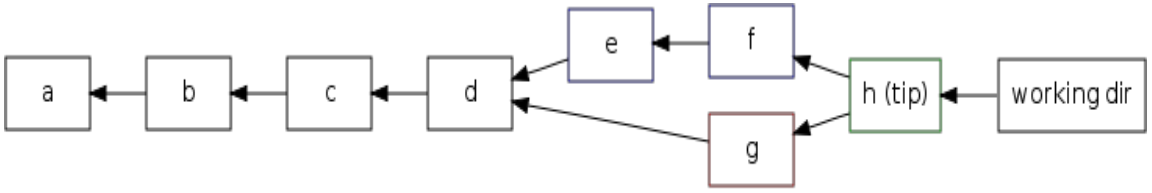
Bob's repo

Bob commituje merge-ovane izmene na svoj lokalni repozitorijum. Sad su njegova radna kopija i repozitorijum poravnati



Alice's repo

Alice fetch-uje izmene sa Bob-ovog repo-a na svoj repo. Njena radna kopija još uvek nije poravnata sa njenim repozitorijumom



Alice's repo

Alice se switch-uje na na Bobove izmene, tako da je njena radna kopija poravnata sa najnovijim izmenama

„Merge“ koncept

- Klasični pristup istovremenog pristupa resursu: „zaključavanje“ pesimistični pristup
- copy-modify-merge pristup – optimistični pristup, koristi se u RCS

Delta Encoding

- Način skladištenja različitih verzija dokumenata putem „delti“
- Skladište se samo uzmene u odnosu na prethodnu verziju fajla
- Prednosti: ušteda skladišnog prostora, lako praćenje izmena

Delta Encoding

Verzija 1:

```
1  public static void main(String[] args){
2      System.out.println(args[0]);
3  }
```

Verzija 2:

```
1  public static void main(String[] args){
2      System.out.println(args[0]);
3      System.out.println(args[1]);
4  }
```

Delta:

```
1      System.out.println(args[1]);
```

Git

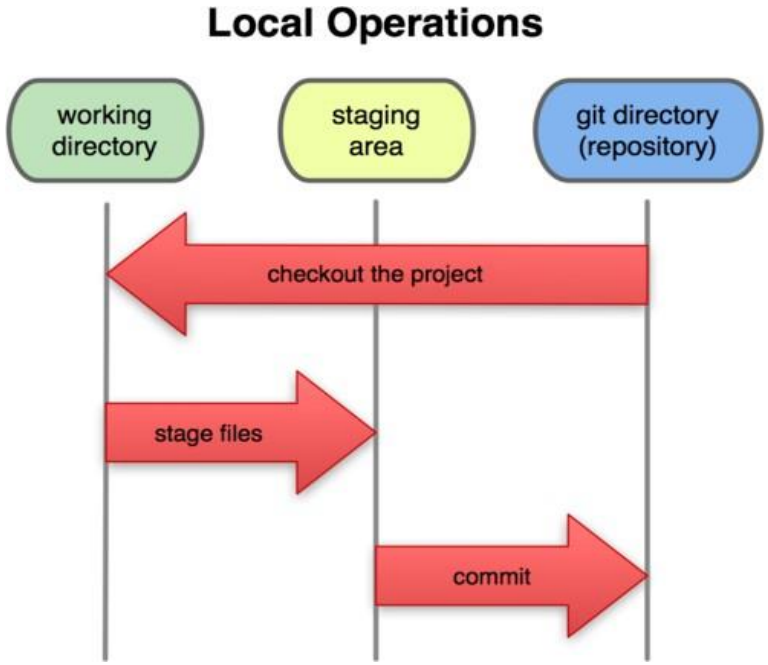
- Kreirao Linus Torvalds, tvorac Linux-a, 2005
- Izašao je iz Linuk razvojne zajednice
- Dizajniran za kontrolu verzija na Linux kernelu

Git – radna kopija

U vašoj lokalnoj kopiji na git-u, datoteke mogu biti:

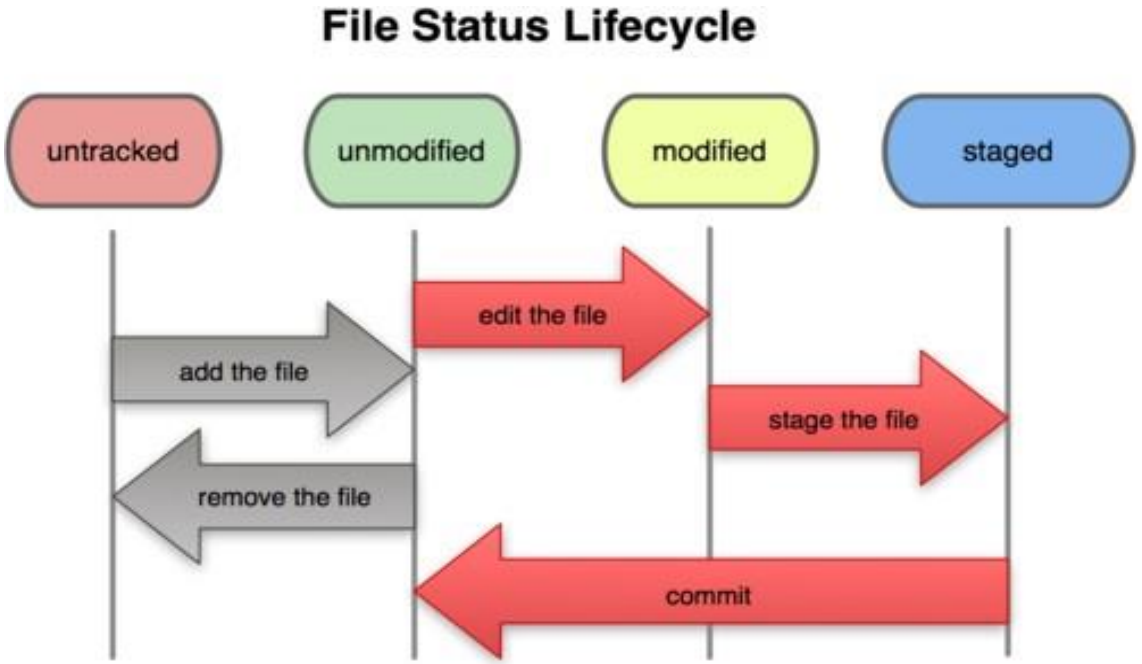
- U vašem lokalnom repo-u (COMMITED)
- Checked out i izmenjeno, ali još uvek ne commited (radna kopija)

Ili - u međustanju ("staging" area)
"Staged" fajlovi su spremni da budu komitovani.



Git – osnovi tok

- **Izmena** fajlova u radnoj kopiji
- Dodavanje (**Stage**) fajlova – dodavanje njihovih snapshot-a u staging oblast.
- **Commit** – preuzima fajlove iz staging oblasti i trajno ih premešta u lokalni Git repozitorijum.



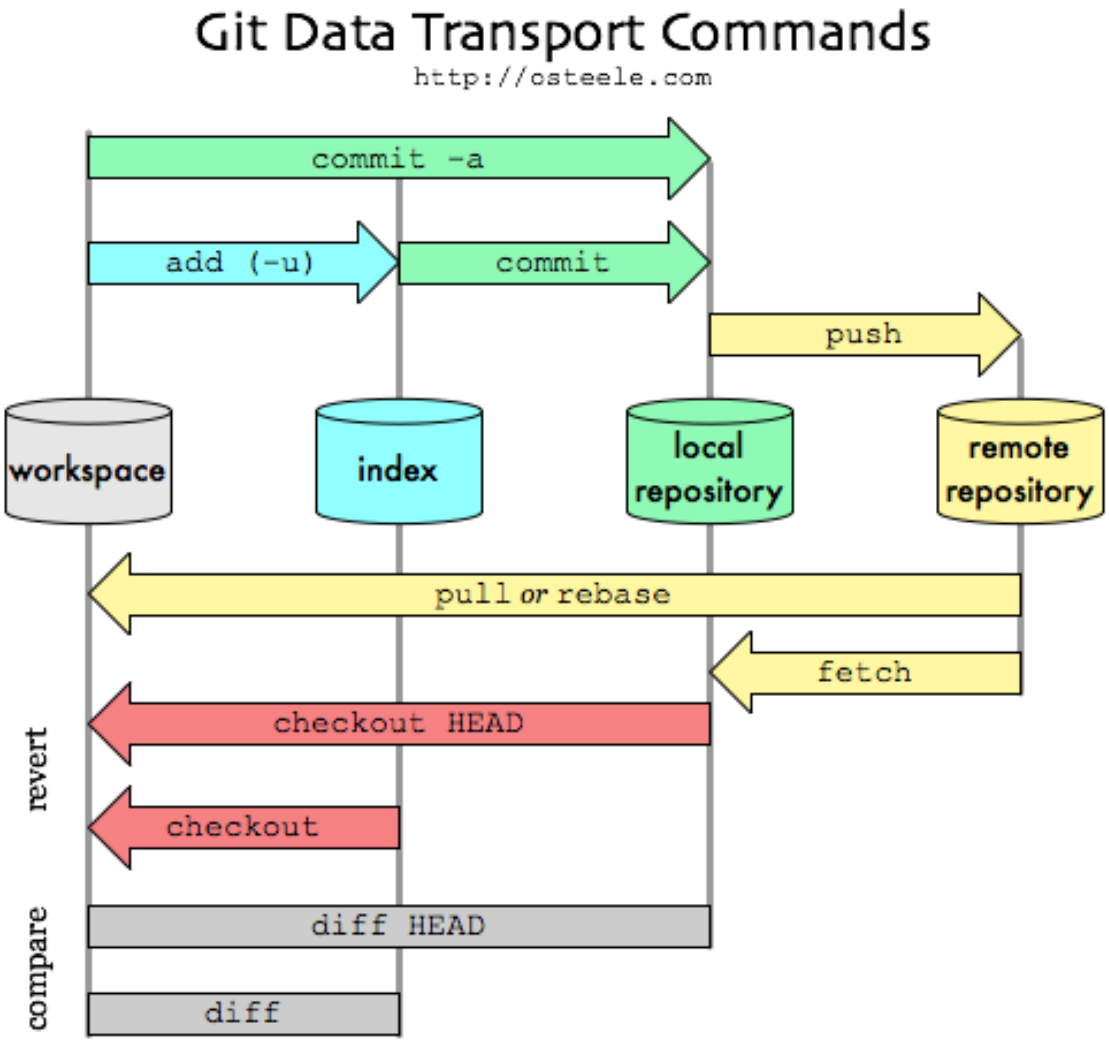
Git – osnovi tok

- U Gitu, svaki korisnik ima sopstvenu kopiju repo-a i commit-uje izmene u svoju lokalnu kopiju repo-a pre push-a na centralni server.
- Git generiše jedinstveni SHA-1 heš (40 znakovnih niza heksadecimalni cifara) za svaki commit.

- Often we only see the first 7 characters:

1677b2d Edited first line of readme
258efa7 Added line to readme
0e52da7 Initial commit

Git


Git Data Transport Commands
http://osteele.com
commit -a
add (-u)
commit
push
workspace
index
local repository
remote repository
pull or rebase
fetch
checkout HEAD
revert
checkout
diff HEAD
compare
diff

Git komande

command	description
`git clone` ***`url [dir]`***	copy a Git repository so you can add to it
`git add` ***`file`***	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help` ***`[command]`***	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init, reset, branch, checkout, merge, log, tag`	

Git - Add and commit a file

- The first time we ask a file to be tracked, and every time before we commit a file, we must add it to the staging area:
  - `git add Hello.java Goodbye.java`
    - Takes a snapshot of these files, adds them to the staging area.
    - In older VCS, "add" means "start tracking this file." In Git, "add" means "add to staging area" so it will be part of the next commit.
- To move staged changes into the repo, we commit:
  - `git commit –m "Fixing bug #22"`
- To undo changes on a file before you have committed it:

Git - Viewing/undoing changes

- To view status of files in working directory and staging area:
  - `git status` or `git status -s` (short version)
- To see what is modified but unstaged:
  - `git diff`
- To see a list of staged changes:
  - `git diff --cached`
- To see a log of all changes in your local repo:
  - `git log` or `git log --oneline` (shorter version)

```
1677b2d Edited first line of readme
258efa7 Added line to readme
0e52da7 Initial commit
```

- `git log -5` (to show only the 5 most recent updates), etc.

Git - Branching and merging

Git uses branching heavily to switch between multiple tasks.

- To create a new local branch:
  - `git branch` *`name`*
- To list all local branches: (* = current branch)
  - `git branch`
- To switch to a given local branch:
  - `git checkout` *`branchname`*
- To merge changes from a branch into the local master:
  - `git checkout master`
  - `git merge` *`branchname`*

Git - Interaction with remote repo

- **Push** your local changes to the remote repo.
- **Pull** from remote repo to get most recent changes.
  - (fix conflicts if necessary, add/commit them to your local repo)

- To fetch the most recent updates from the remote repo into your local repo, and put them into your working directory:
  - `git pull origin master`

- To put your changes from your local repo in the remote repo:
  - `git push origin master`

Izvori, literatura

- https://betterexplained.com/articles/a-visual-guide-to-version-control/
- https://www.tutorialspoint.com/git/index.htm

Hvala na pažnji