nedelja informatike 3

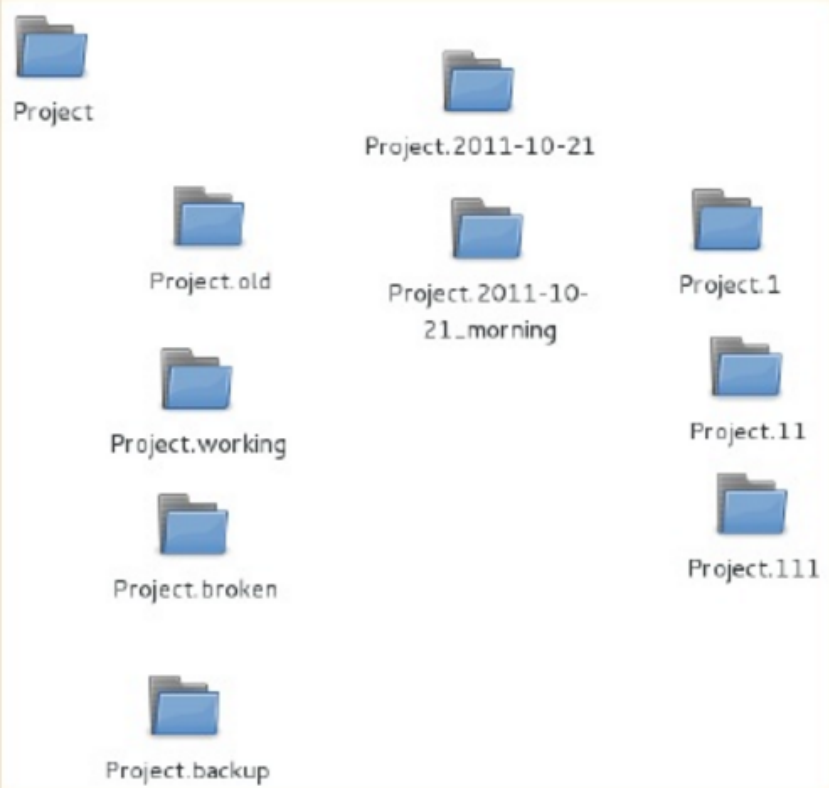
git: distribuirana kontrola verzije softvera

Nikola Jovanović

Matematička gimnazija
Nedelja informatike 3

12. decembar 2016.

Šta je kontrola verzije softvera?



Zašto je korisna?

- Dodaje na sigurnosti čuvanjem prethodnih verzija
- Pomaže da pratimo sopstvene promene (i eventualno ih poništimo)
- Olakšava eksperimentisanje
- Omogućava kolaborativni razvoj

Pristupi

- Centralizovani
  - CVS, SVN, Perforce
- Distribuirani
  - Mercurial, Bazaar, Git

git

- Autor: Linus Torvalds (2005)
- Nastao za potrebe razvoja Linux kernela
- Danas najpopularniji alat u svojoj kategoriji
  - Open-source projekti
  - Velike kompanije
  - Integracija sa razvojnim okruženjima
  - Web source-code hosting sajtovi (GitHub $\neq$ Git)

Rad sa git-om

- Rad iz komandne linije
  - https://git-scm.com/
- GUI okruženja
  - git-gui, gitk
  - Github Desktop, Tower

Osnovni koncepti

- **Repozitorijum** - struktura koja čuva istoriju projekta
  - Čuvaju se snimci (snapshots) umesto svake zasebne edit operacije
  - Aciklični usmereni graf commit-ova

- **Commit** - „snimak”, odgovara stanju projekta u jednom trenutku vremena
  - Sadrži pokazivače ka roditeljskim commit-ovima, datum stvaranja, podatke o autoru i opis
  - Svaki commit ima svoj **globalno** jedinstveni ID - 160-bitni SHA heš

Grananja

- **Branch** - jedna od imenovanih linija razvoja
  - Glave grana se kreću zajedno sa novim commit-ovima
  - Default grana: master
  - Kreiranje nove grane $\Longrightarrow$ commit sa više dece
- **Merge** - stapanje dve ili više grana u jednu
  - Strategije, fast-forward, konflikti
  - Merge grana $\Longrightarrow$ „merge commit” sa više očeva

Još neki objekti

- **HEAD** - simbolička veza, pokazuje na trenutno aktivnu granu
- **Checkout** - promena aktivne grane, pomera HEAD
- **Tag** - opisno ime za određeni commit
  - Primer: release-2.4.7
  - Tagovi uglavnom ne menjaju commit na koji pokazuju
- Moguć je checkout na tag tj. na konkretan commit pri čemu se ulazi u **detached head** mod: novi commit-ovi pomeraju HEAD, ali na tom mestu nema grane
  - Ako se HEAD pomeri commit-ovi ostaju da „vise”

Storage oblasti

- **repozitorijum** - opisana baza podataka koja čuva graf commit-ova, glave grana, tagove, HEAD
- **radni direktorijum** - mesto gde modifikujemo fajlove našeg projekta
- **index (staging area)** - oblast za manevrisanje, odvojena od radnog direktorijuma, gde se prenose fajlovi koje pripremamo za dodavanje novog commita
- **stash** - storage oblast u koju možemo privremeno smeštati work-in-progress objekte iz radnog direktorijuma (štek)
- index, repo i stash se čuvaju u skrivenom folderu `.git/`

Kolaboracija

- Repozitorijumi su dostupni preko URL (lokalno/web)
- **remote** - udaljeni repozitorijum koji pratimo (jedan ili više)
  - remote grane imaju specijalna imena oblika **remote1/grana1**
- **origin** - specijalno ime za repozitorijum koji smo klonirali pri kreiranju trenutnog

Kreiranje repozitorijuma, staging

- `git init` - kreira prazan repozitorijum (/.git direktorijum)
- `git add putanja/fajl.txt` - dodaje fajl iz radnog direktorijuma u indeks
  - `git add .` - dodaje sve modifikovane fajlove u indeks
- `git status` - izlistava fajlove u indeksu i modifikovane fajlove u radnom direktorijumu

Commit

- `git commit` - čuva trenutnu sadržinu indeksa u vidu novog commit-a na trenutnoj grani, po default-u otvara podešeni editor za upisivanje commit poruke
  - `git commit -m ,,poruka''` - dozvoljava upis commit poruke iz konzole
- Postavljanje detalja o autoru:
  - `git config -global user.name ,,Nikola Jovanovic''`
  - `git config -global user.email ,,randomusername@csnedelja.mg.edu.rs''`

Grananje

- `git branch` - izlistava sve lokalne grane (markira HEAD sa *)
  - `git branch -a` - izlistava sve lokalne i remote grane
  - `git branch bugfix1` - započinje novu granu pod imenom bugfix1 (počinje od HEAD)
  - `git branch bugfix1 commit` - započinje novu granu pod imenom bugfix1 koja počinje od navedenog commit-a
  - `git branch -d bugfix1` - briše granu bugfix1

Manipulacije stablom

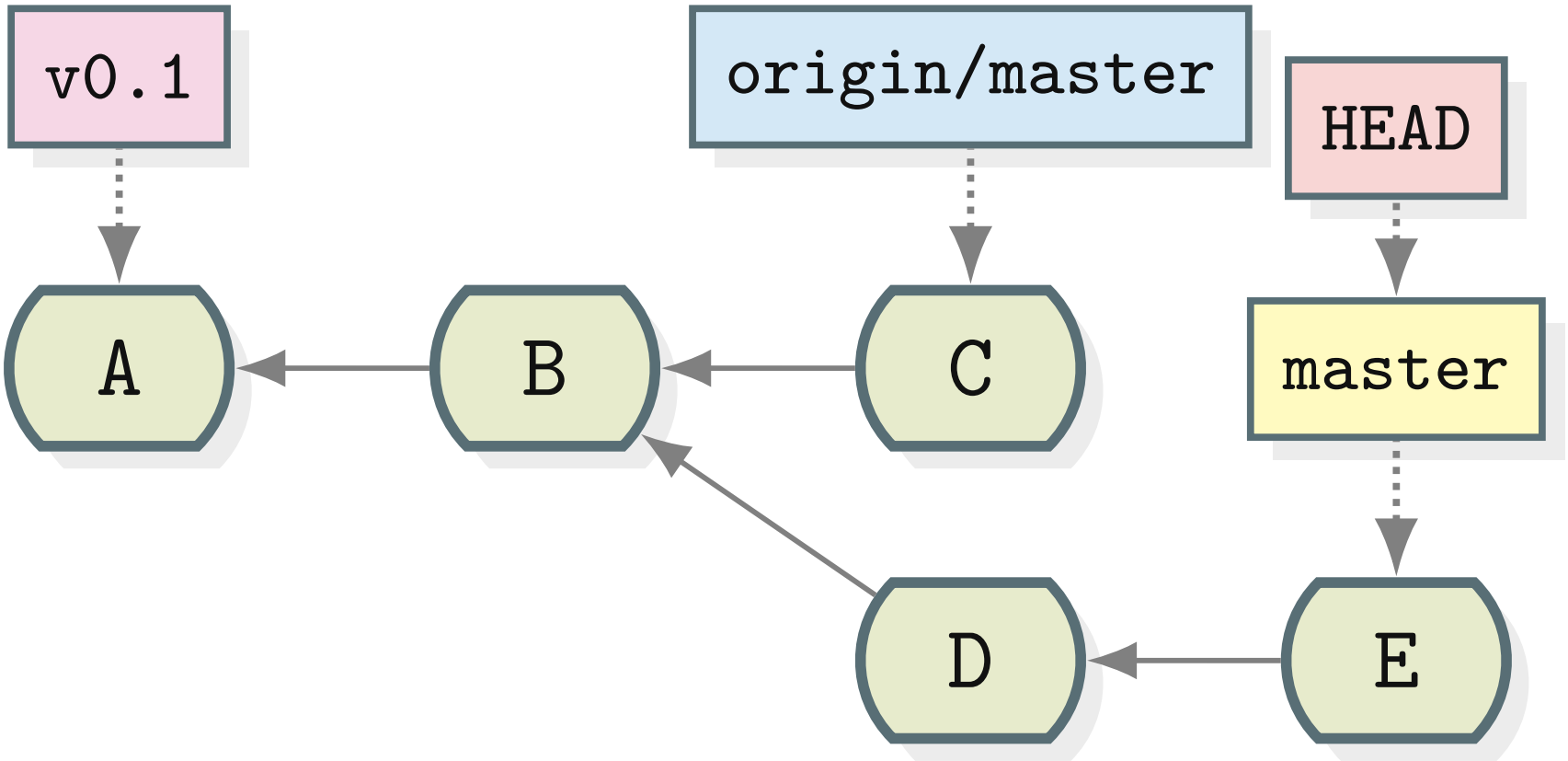
- `git merge bugfix1` - stapa granu bugfix1 sa trenutno aktivnom granom
  - moguće je stapanje više grana odjednom
- `git tag 1.0.0 commit` - kreira novi tag koji pokazuje na navedeni commit
- `git checkout bugfix1` - grana bugfix1 postaje aktivna (prebacuje se HEAD na nju) i ažurira se sadržaj radnog direktorijuma i indeksa prema sadržaju te grane
  - priprema za rad na toj grani, novi commit-ovi će se dodavati na nju
  - `git checkout -b bugfix1` - kreira granu bugfix1 i radi checkout na nju
  - `git checkout fajl.txt` - vraća filename u radnom direktorijumu na verziju koja je u HEAD

Ispravljanje grešaka

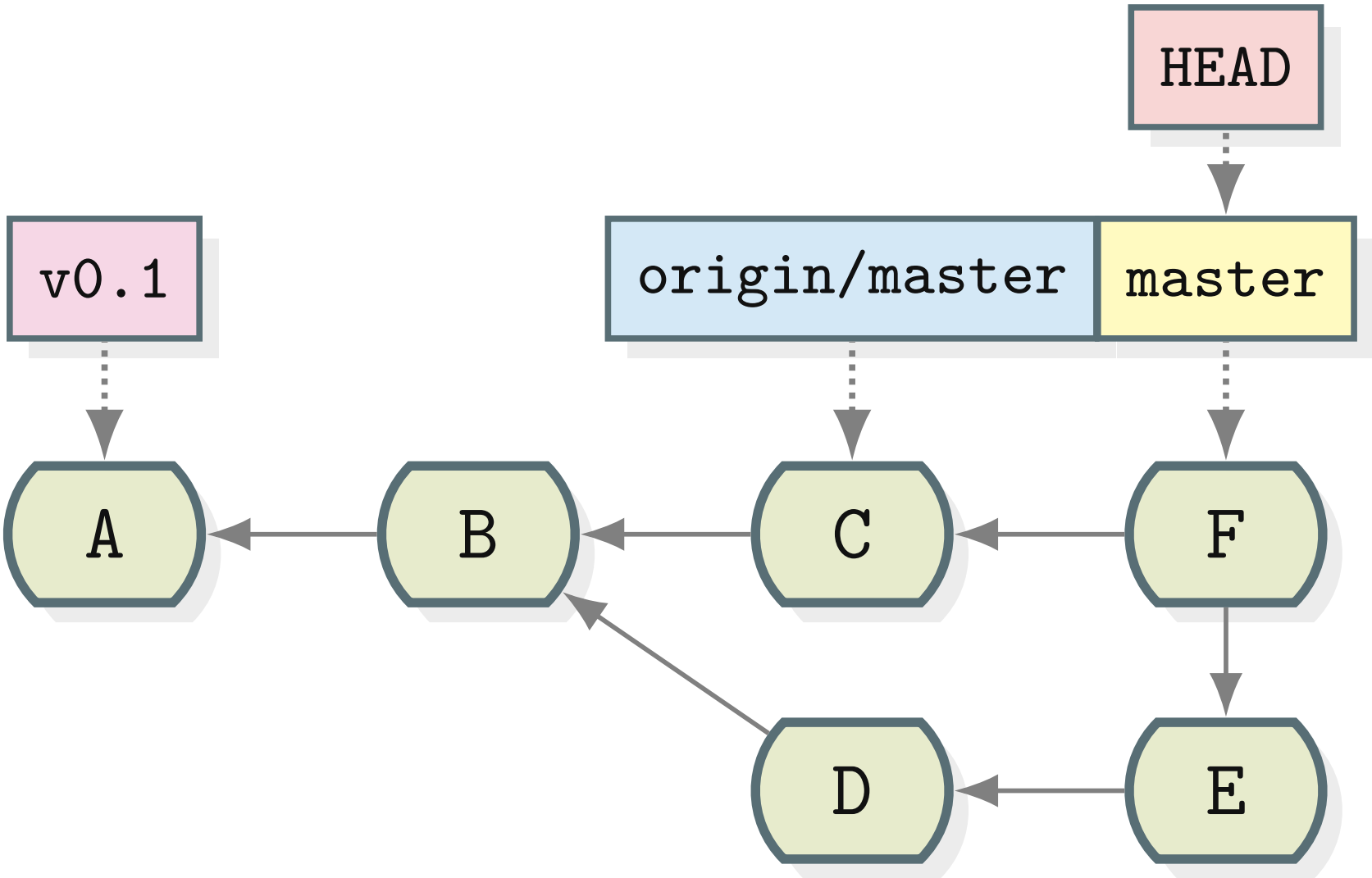
- `git reset`
  - `--soft HEAD~` - uklanja poslednji commit
  - `--mixed HEAD~` - soft + ažurira index
  - `--hard HEAD~` - mixed + ažurira radni direktorijum

- `git rebase commit` - menja istoriju trenutne grane tako da je odabrani commit nova početna tačka
- Obratimo pažnju na glavne razlike između `merge` i `rebase`...

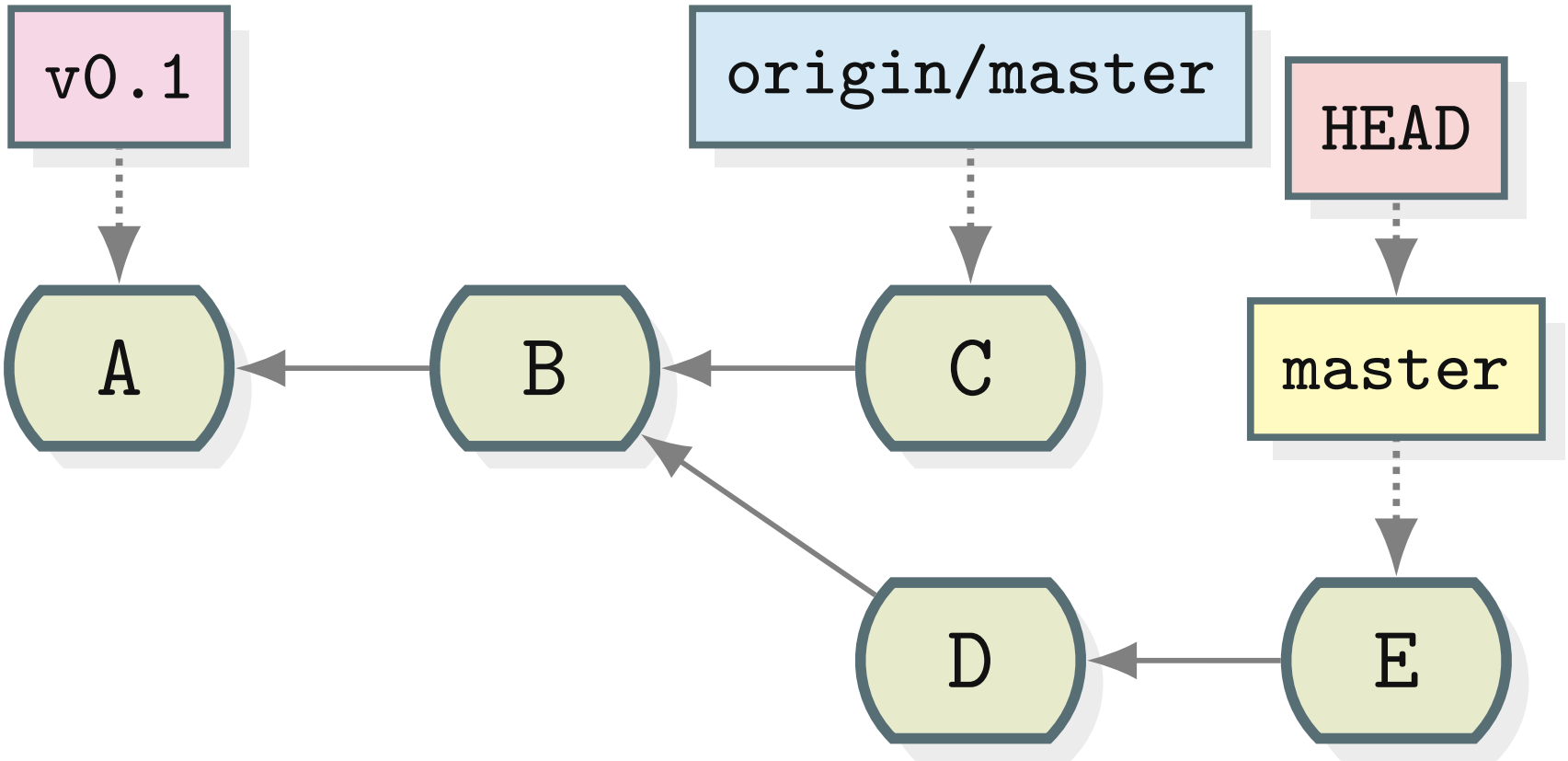
Primer repozitorijuma



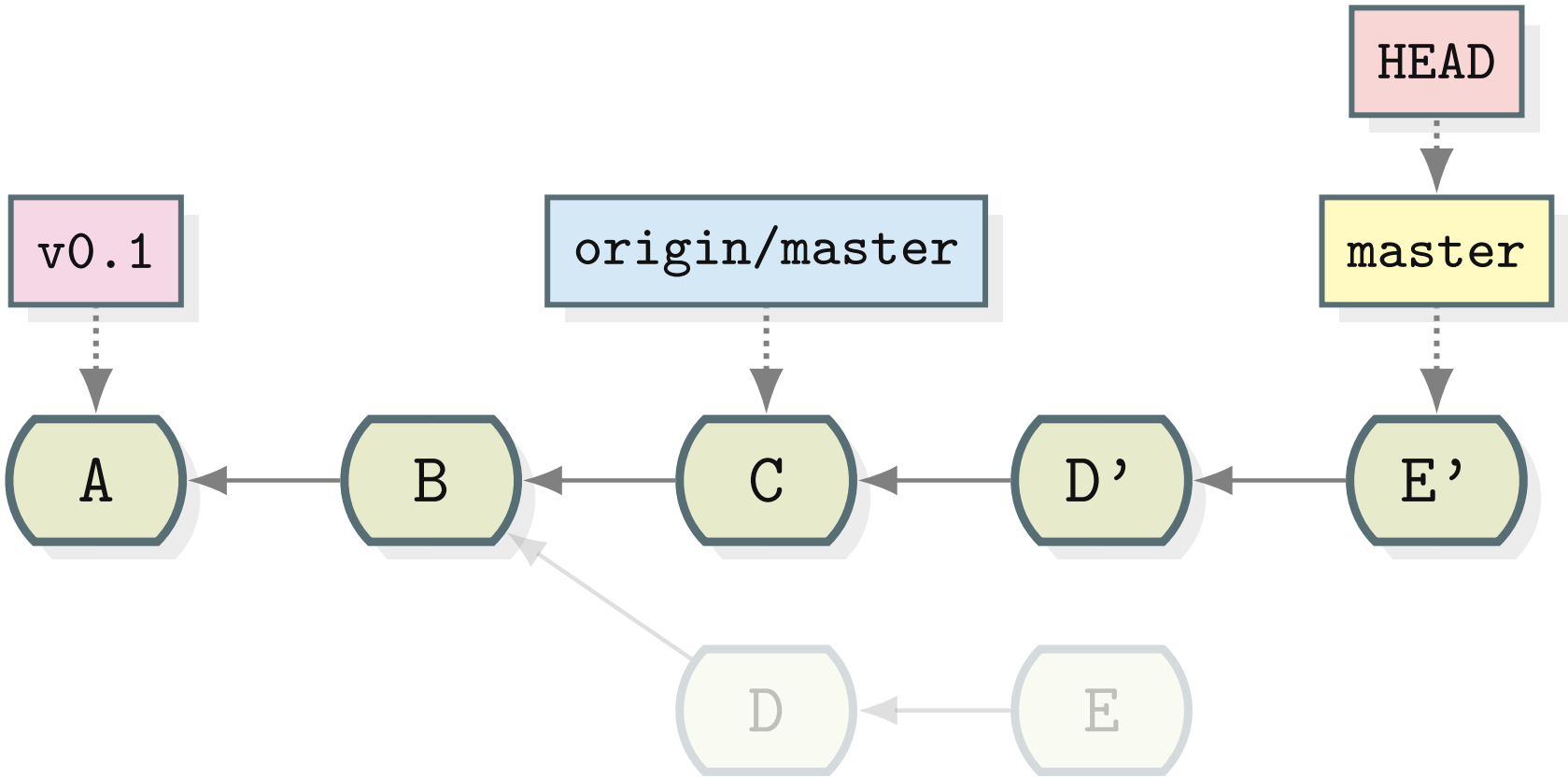
Repozitorijum nakon merge



Primer repozitorijuma



Repozitorijum nakon rebase



Budite pažljivi!

You have been warned

Ukoliko su naše kolege već pokupile pogrešni commit koji smo napravili, uglavnom je bolje da grešku popravimo u novom commitu umesto da menjamo istoriju. U suprotnom, svi kolaboratori koji su pokupili našu grešku će morati da rade `git rebase` svojih grana na naš popravljen commit!
Menjanje istorije nije preporučljivo upravo iz ovih razloga!

Kolaboracija

- `git clone username@host:/url/repo` - kopira udaljeni repozitorijum i pravi radni direktorijum oko njega; izvorni repozitorijum biva zapamćen u kloniranom pod imenom `remote`
- `git fetch` - preuzima promene iz udaljenog repozitorijuma (default: origin) ali ih ne stapa u radni direktorijum
- `git pull` - preuzima promene iz udaljenog repozitorijuma (default: origin) i stapa ih u radni direktorijum
  - pull = fetch + merge
- `git push` - prosleđuje lokalne promene u udaljeni repozitorijum

Još neke komande

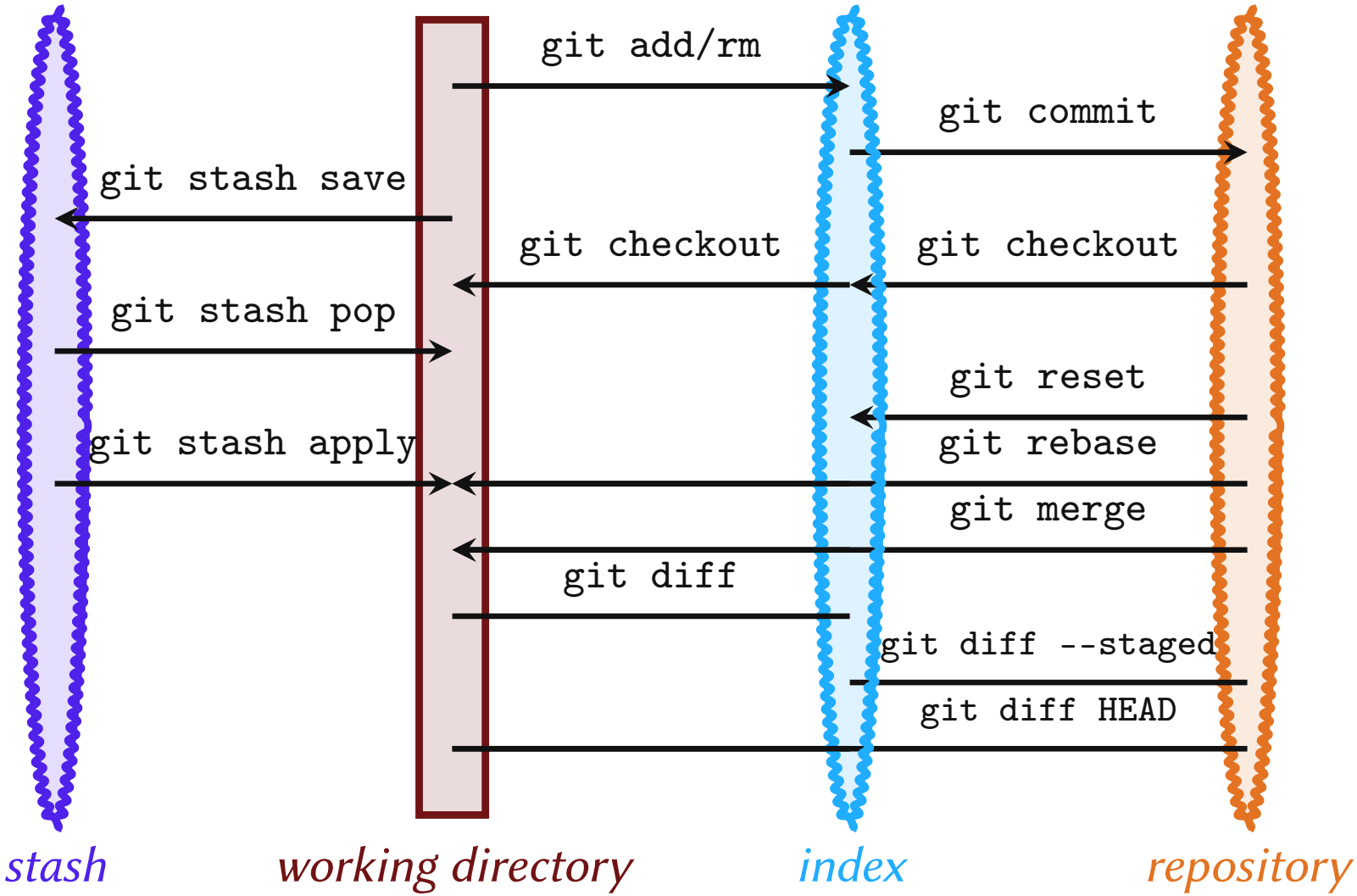
- `git diff` - prikazuje razlike između radnog direktorijuma i indeksa
  - `git diff HEAD` - poredi radni direktorijum i HEAD
  - `git diff --staged` - poredi indeks i HEAD
  - `git diff grana1 grana2` - prikazuje razlike između dve grane
- `git stash` - prebacuje sadržaj radnog direktorijuma u stash
- `git stash pop` - vraća fajlove iz stash-a u radni direktorijum

Još neke komande (2)

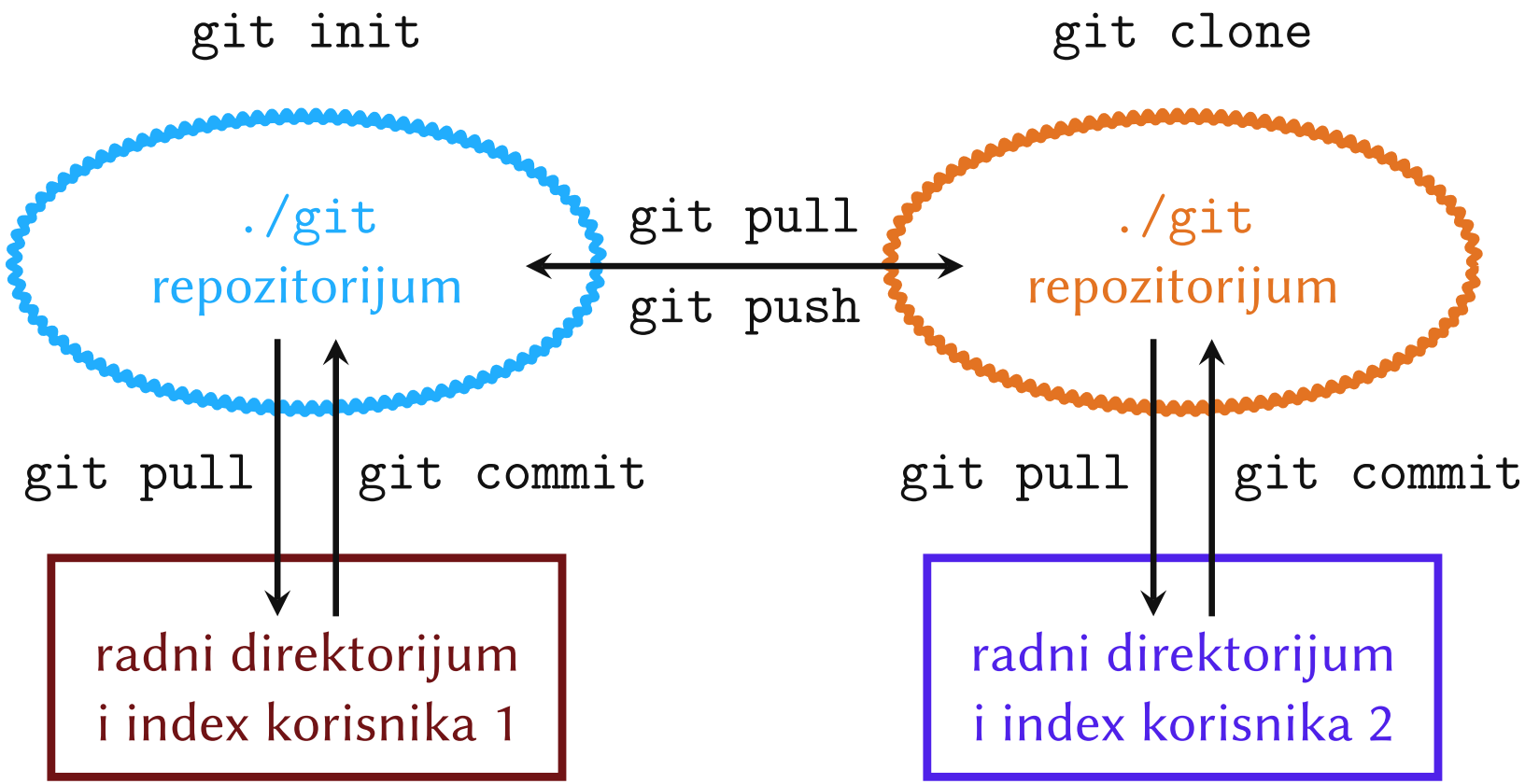
- `git log` - prikazuje informacije o prethodnim commit-ovima
  - moguće je prikazati promene za svaki commit, kao i filtrirati commit-ove po datumu, autoru, promenjenim fajlovima, poruci...
  - postoji veliki broj flag-ova koji olakšavaju tumačenje log poruke
  - primer: `git log --graph --oneline --decorate --all` - prikazuje istoriju celog repozitorijuma u obliku ASCII grafa pri čemu svaki commit zauzima jednu liniju i prikazuju se odnosi sa granama i tagovima

- i još mnoooogo njih...

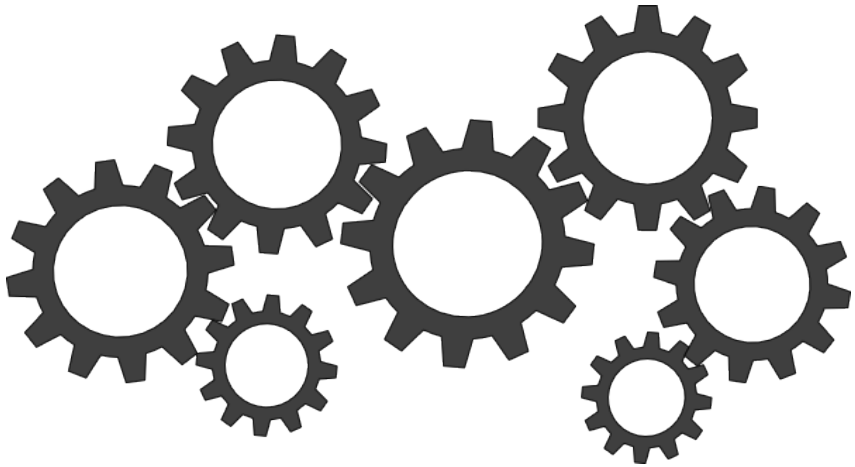
git – protok informacija između storage oblasti



git – protok informacija između storage oblasti



Demo



Linkovi

- Zanimljiv uvod u git: http://gitolite.com/gcs.html
- Interaktivni tutorijal: http://learngitbranching.js.org/
- Pregledan cheat sheet: https://education.github.com/git-cheat-sheet-education.pdf

Hvala na pažnji!