



Praktikum iz operativnih sistema 1

Vežbe 7

Oblasti

- git

git

- uvod -

- `git` je besplatan alat za verzionisanje koda
- dizajniran i napravljen 2005. od strane Linux zajednice
- `git` čuva sve fajlove projekta i njegovu istoriju menjanja
 - Svaka verzija projekta predstavlja jedan `commit`
 - Korisnik ručno dodaje komite
 - Može se pristupiti bilo kom komitu, tj. verziji projekta
- `git` repozitorijum je folder u kome se čuva projekat, komiti i grane
 - Repozitorijumi se lako mogu deliti sa ostalim korisnicima preko host servisa kao što su `Bitbucket`, `GitHub`, `Codebase`...

Najnoviji commit je na vrhu!

Svaki commit zna svog prethodnika

Svaki commit pripada nekoj grani

Svaka tačka predstavlja jedan commit

git

- primer stabla -

Svaki commit ima svoj opis i vreme stvaranja

```
* 525a852 - Sun, 29 Jul 2012 13:04:46 -0700 (9 months ago) (refs/remotes/origin/test_on_0.8)
notification test - German Laullon
* 58b5a57 - Fri, 27 Jul 2012 00:18:16 -0700 (9 months ago)
morden Objective-c - German Laullon
* aef1d39 - Fri, 27 Jul 2012 00:07:45 -0700 (9 months ago)
first build. - German Laullon
* bff6661 - Wed, 25 Jul 2012 10:24:23 -0700 (9 months ago) (HEAD, refs/remotes/origin/master, refs/remotes/origin/HEAD)
Merge pull request #169 from taybin/patch-1 - German Laullon
* ef61b97 - Mon, 12 Mar 2012 23:14:01 -0300 (1 year, 2 months ago)
Update Rakefile to work with ruby-1.9.2. - Taybin Rutkin
* 3b06317 - Wed, 25 Jul 2012 10:23:49 -0700 (9 months ago)
Merge pull request #175 from barrywardell/master - German Laullon
* 907e967 - Wed, 1 Feb 2012 10:23:38 +0000 (1 year, 3 months ago)
Fix bug where submodules were incorrectly grouped when the first part of their path was the same. - Barry Wardell
* ddb1b324 - Wed, 25 Jul 2012 10:23:08 -0700 (9 months ago)
Merge pull request #195 from jphalip/master - German Laullon
* 8ebb58c - Mon, 18 Jun 2012 22:42:33 -0700 (11 months ago)
Added "Copy Reference to Clipboard" context menu item in sidebar. - Julien Phalip
* 238a97a - Sat, 16 Jun 2012 21:19:58 -0700 (11 months ago)
Fixed a tiny typo. - Julien Phalip
* 3386fc7 - Sat, 16 Jun 2012 21:18:14 -0700 (11 months ago)
Make sure the commit view gets refreshed when 'Stage' gets selected and the auto-refresh is on. - Julien Phalip
* 7fafdb8 - Sun, 10 Jun 2012 13:36:48 -0700 (11 months ago)
Tweaked capitalization of words in contextual menu items. - Julien Phalip
* 5958f5b - Sun, 10 Jun 2012 13:24:46 -0700 (11 months ago)
Don't display all the files selected for deletion to avoid the confirmation sheet getting too tall if there are many files. - Julien Phalip
* 3575e87 - Sat, 9 Jun 2012 15:37:40 -0700 (11 months ago)
Prevent large files from getting loaded in the commit view to prevent the app from freezing. - Julien Phalip
* 748621c - Fri, 8 Jun 2012 12:10:38 -0700 (11 months ago)
Made the "(Un)stage lines" functionality in the commit view work only if the Command key is pressed. - Julien Phalip
* d249fd6 - Thu, 7 Jun 2012 19:41:30 -0700 (11 months ago)
When a branch gets selected in the sidebar, make sure its corresponding commit also gets selected in the commit view. - Julien Phalip
* 839c9b6 - Thu, 7 Jun 2012 19:36:56 -0700 (11 months ago)
Made the diff table adapt nicely to the window's size. Fixes #50. - Julien Phalip
* 8badd8a - Wed, 25 Jul 2012 10:21:54 -0700 (9 months ago)
Merge pull request #196 from Kyriakis/master - German Laullon
* af773ba - Sat, 9 Jun 2012 08:09:55 +0200 (11 months ago)
No check for refs on remotes while deleting them - Robert Kyriakis
* 47a8a1a - Wed, 25 Jul 2012 10:21:14 -0700 (9 months ago)
Merge pull request #197 from Uncommon/arcfix - German Laullon
* d1a8ba9 - Fri, 15 Jun 2012 11:18:13 -0600 (11 months ago)
Fix for #196 - Uncommon/arcfix - Uncommon
```

Svaki commit ima svoj hash kod kao id, da bismo mogli da ga referenciramo kada budemo želeli

git

- config, init, clone, pull -

- Postavljanje imena i mejl adrese kako bi se znalo ko je pravio koji commit

```
$ git config --global user.email "mejl"
$ git config --global user.name "ime"
```
- Pravljenje lokalnog repozitorijuma "test"

```
$ mkdir test && cd test
$ git init
```
- Dodavanje lokalnog repozitorijuma u prazan udaljeni repozitorijum

```
$ git remote add origin <repo_url>
$ git push -u origin master
```
- Kloniranje repozitorijuma sa host servisa

```
$ git clone <repo_url>
```
- Sinhronizovanje lokalne grane "grana" sa udaljenom granom "grana"

```
$ git checkout grana
$ git fetch origin grana
$ git merge origin/grana
- III
$ git checkout grana
$ git pull grana
```

git

- stage, commit, push -

- Interaktivno dodavanje ili uklanjanje fajlova/izmena novom komitu (staging)
\$ `git add -i`
- Dodavanje svih izmena novom komitu (staging)
\$ `git add --all`
- Pravljenje komita nakon prethodnog staging koraka
\$ `git commit -m "komentar"`
- Ažuriranje poslednjeg komita (na vrhu grane)
\$ `git commit -amend`
 - Nano editor, pamćenje i izlazak: CTRL+x, y, enter
- Ažuriranje grane "grana" udaljenog repozitorijuma
\$ `git push origin grana`

git

- brisanje ne komitovanih izmena -

- Ukoliko želimo da obrišemo neke **ne komitovane** izmene tako da izgleda da se nikada nisu ni desile, postoje 3 slučaja:

- untracked changes – ukloni fajl

```
$ git clean -n // šta bi clean -f uradio
```

```
$ git clean -fd // ukloni sve nove fajlove i foldere
```

- unstaged changes – ukloni fajl

```
$ git checkout -- fajl.txt
```

- staged changes – ukloni fajl/ove

```
$ git reset fajl.txt
```

```
$ git reset HEAD
```

git

- grane -

- Pravljenje, odnosno brisanje lokalne grane "grana"
 - \$ `git branch grana`
 - \$ `git branch -d grana`
 - **Grana je, u suštini, samo referenca na komit!**
- Istovremeno pravljenje i biranje grane "grana"
 - \$ `git checkout -b grana`
- Biranje postojeće grane "grana"
 - \$ `git checkout grana`
 - Biranjem grane u repozitorijumu se pojavljuje verzija koda zapamćena u komitu na koji pokazuje grana
- Pokaži sve lokalne grane u repozitorijumu
 - \$ `git branch`
- Pokaži sve komite vezane za granu1
 - \$ `git log grana1`

git:

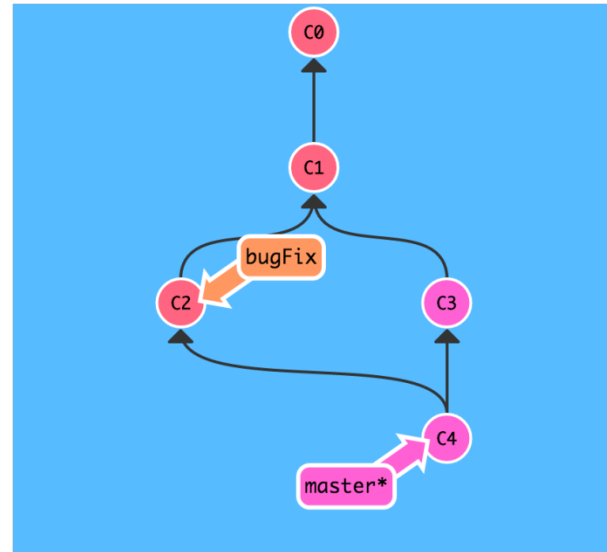
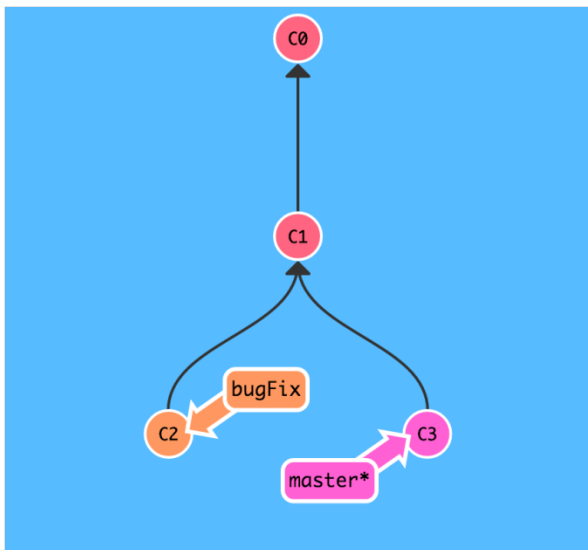
- spajanje grana -

- Spajanje grane `bugFix` u granu `master`

\$ `git checkout master`

\$ `git merge bugFix`

- `bugFix` grana će da se spoji sa tekućom, `master` granom, čime će da se dobije novi komit C4, tj. verzija koja sadrži uniju poslednjeg stanja iz obe grane
- **Prisetiti da je slika stabla ovde okrenuta naopačke, tj. najnoviji komit je skroz dole (samo zapaziti orijentaciju strelica)**



git

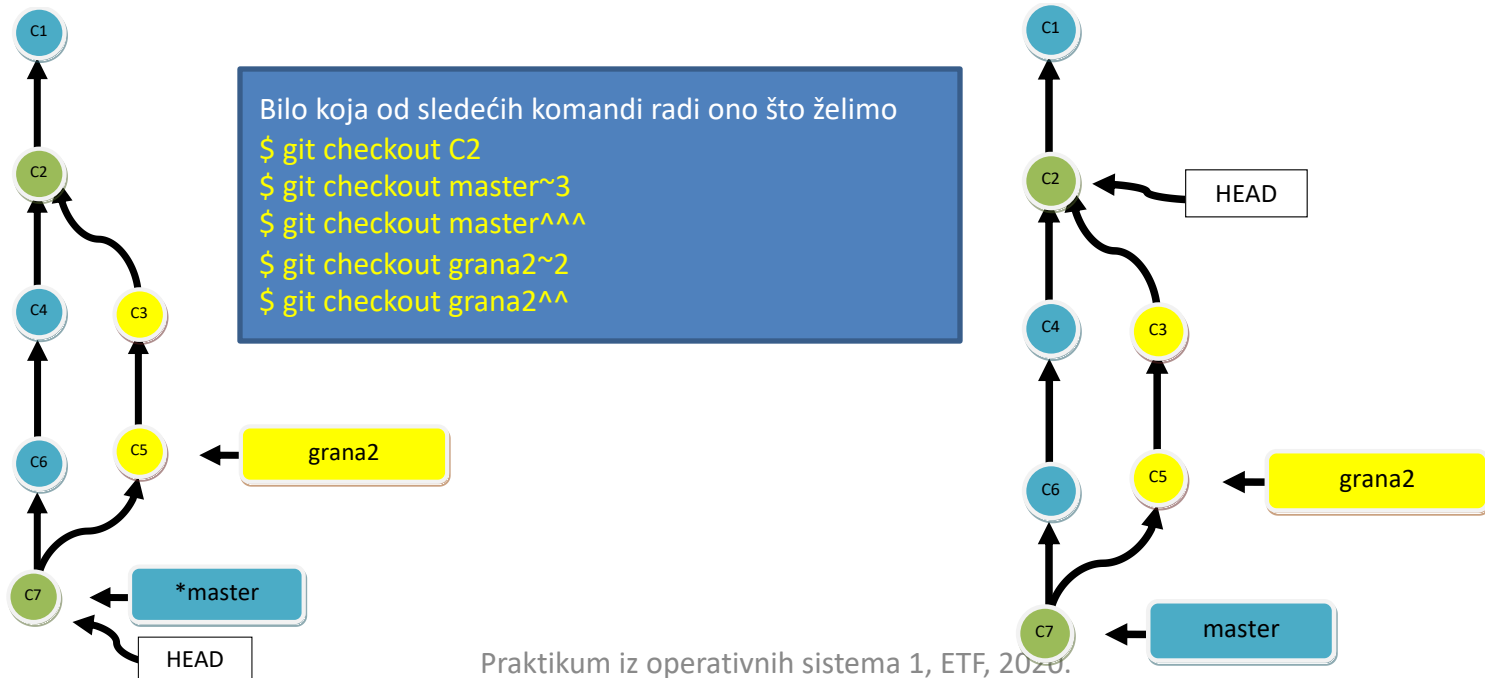
- šetenja kroz istoriju -

- Da bismo mogli da se šetamo kroz verzije koda koristimo referencu `HEAD`
- `HEAD` je simboličko ime za tekući komit, odnosno tekuću verziju koda
- Postoje više načina da se premestimo na novi komit
 - Apsolutno
 - \$ `git checkout hesh_tag_komita`
 - Relativno (2 načina)
 - \$ `git checkout grana^`
 - ili
 - \$ `git checkout grana~1`
 - `grana^` i `grana~1` pokazuju na roditeljski komit onog komita na koji ukazuje `grana`
 - Napomena – `HEAD` će biti u *detached* stanju, tj. nećemo biti na konkretnoj grani. Da bismo se vratili na neku granu, `git checkout grana`

git

- šetnja kroz istoriju: primer -

- Recimo da je tekuće stanje kao na slici levo (**poslednji komit je na vrhu**)
 - Trenutno smo u master grani na komitu sa hesh tagom C7
 - To znači da trenutno gledamo verziju koda u komitu C7
- Recimo da želimo da vidimo verziju projekta u komitu C2, slika desno
 - Potrebno je da HEAD premestimo na C2



git

-grana vs HEAD -

- Ne mešati grane i HEAD referencu
 - **HEAD je jedan i to je referenca na tekući komit**
 - Verzija koda koja se gleda je određena tekućim komitom
 - Grana može da ima puno i to su opet pokazivači na određene komite
 - Pomoću grana možemo da pravimo nove komite, radimo push, pull, merge... One oblikuju stablo commita.
 - Kada odaberemo `granu1` sa `git checkout granu1`, HEAD automatski pokazuje na isti komit kao i ta `granu1`
- Grane se mogu premeštati preko HEAD pokazivača
 - \$ `git branch -f grana HEAD~1`
 - `grana` će da se premesti da pokazuje na roditeljski komit tekućeg komita

git

- prikaz stabla -

- Prikaz stabla

- U fajl ~/.gitconfig uneti koda dole, a nakon toga izvršiti sledeću komandu

- \$ git lg1

- ili

- \$ git lg2

[alias]

```
lg1 = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' --all
```

```
lg2 = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold yellow)%d%C(reset)%n" %C(white)%s%C(reset) %C(dim white)- %an%C(reset)' --all
```

```
lg = !"git lg1"
```

```
c2d8c3c Tue, 12 Apr 2016 13:17:21 +0200 (19 seconds ago) (HEAD, master)
|
|   master:   Dodati tekst u test - TestUser
|   901f8ba Tue, 12 Apr 2016 13:15:42 +0200 (2 minutes ago) (grana2)
|   |
|   |   grana2: Dodali tekst u test2 - TestUser
|   |   4b9beae Tue, 12 Apr 2016 13:14:19 +0200 (3 minutes ago)
|   |   //
|   |   grana2: Napravljen fajl test2 - TestUser
|   8bc2afb Tue, 12 Apr 2016 13:12:44 +0200 (5 minutes ago)
|   |
|   |   master: Napravljen fajl test1 - TestUser
```

Rezultat izvršavanja
\$ git lg2

- Prikaz svih komita tekuće grane

- \$ git log

- Prikaz svih komita grane "grana"

- \$ git log grana

- Opcija --oneline za prikaz komita u jednoj liniji

- Prikaz svih lokalnih grana

- \$ git branch

git

- brisanje komita-

- **Brisanje LOKALNOG komita**

- Ako su izmene bile lokalne, koristiće se jedna od sledeće dve linije

- \$ git reset --hard HEAD~1

- ← --hard opcijom se brišu izmene iz brisanog komita

- \$ git reset --soft HEAD~1

- ← --soft opcijom se NE brišu izmene iz brisanog komita

- Tekuća grana odlazi na prethodni komit i briše onaj na koji je ukazivala pre toga

- Brisanje svih lokalnih promena

- \$ git reset --hard HEAD

- Ne utiče na *untracked* delove

- **Brisanje UDALJENOG komita**

- Brisanje komita koji je već na serveru

- git revert HEAD – dobio bi se novi komit kojim se beleži taj korak unazad
 - git push --force origin - poslednji komit na serveru se briše ovim novim
 - ili ako se zeli totalno brisanje, onda bez prve komande

git

- ažuriranje komita -

- **Ažuriranje LOKALNOG komita**

- Ako je komit i dalje lokalan **i na vrhu grane**, koristiće se

- \$ **git commit --amend**

- Tekućem komitu se na ovaj način menjaju izmene, poruka...I NE PRAVI SE NOVI KOMIT.

- **Ažuriranje UDALJENOG komita**

- Ako je komit već na serveru, koristiće se kombinacija

- \$ git commit --amend

- \$ git push --force-with-lease origin tekuca_grana

- Prvom komandom se dobio novi komit koji se nastavlja u lokalnoj grani tekuca_grana (dešava se račvanje sa origin/tekuca_grana)
 - Drugom komandom uništavamo poslednji komit udaljene grane tekuca_grana i zamenjujemo ga poslednjim komitom lokalne grane tekuca_grana
 - DAKLE, NEMA NOVIH KOMITA, SAMO SE ZAMENIO STARI

- **Bitna napomena: Poslednje neće raditi ukoliko je neko uradio pull pre nego što smo mi radili ovaj nasilni push**

Open source software



Open source software

- Principi otvorenog softvera (eng. Open source software)
 - Dostupnost izvornog koda bilo kome
 - Korišćenje koda na bilo koji način
 - Menjanje koda na bilo koji način
 - Deljenje koda bilo kome u bilo koju svrhu
 - I sve to u skladu sa licencom!
- Tipovi licence (grubo podeljeni)
 - Give me credit (Apache, OpenBSD...)
 - Share fixes with me (GNU Lesser General Public Licence, Mozilla licence, Eclipse licence...)
 - Give me everything (GPL)

Open source: Zašto ga koristiti?

- Danas više od 180 000 projekata
- Prednosti open source softvera
 - Niža cena
 - Bezbednost
 - "Given enough eyeballs, all bugs are shallow" nasuprot "Security through obscurity"
 - Bagovi se brže uočavaju i popravljaju
 - Otvorenost koda
 - Doprinos učenju gledanjem tuđeg koda i menjanje koda po potrebi
 - Bolji kvalitet i prilagodljivost
 - Bliže je onome što korisnik želi, zato što ga korisnik i i menja

Open source: Ko piše to i zašto?

- Kompanije
 - Softver koji je u širokoj upotrebi
 - Unapređenje koda delegiranjem posla
 - Zajedica koja radi na istom cilju
 - Smanjenje troškova
- Volonteri, studenti
 - Sjajan način za profesionalno napredovanje
 - Mentorisan rad
 - Procesi u projektu i timu
 - *Code review*
 - Učenje analizom tuđeg koda
 - Odlična preporuka za posao
 - Prilika za zaradu
(na određenim projektima i u skladu sa stečenom reputacijom)

Open source: Primeri

- Neki projekti posebno aktuelni 2015. godine:

- [Apache Spark](#)



- [Blender](#)



- [D3](#)



- [git](#)



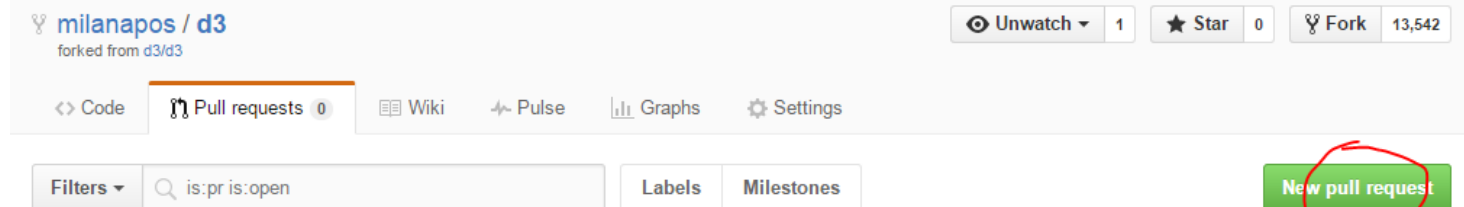
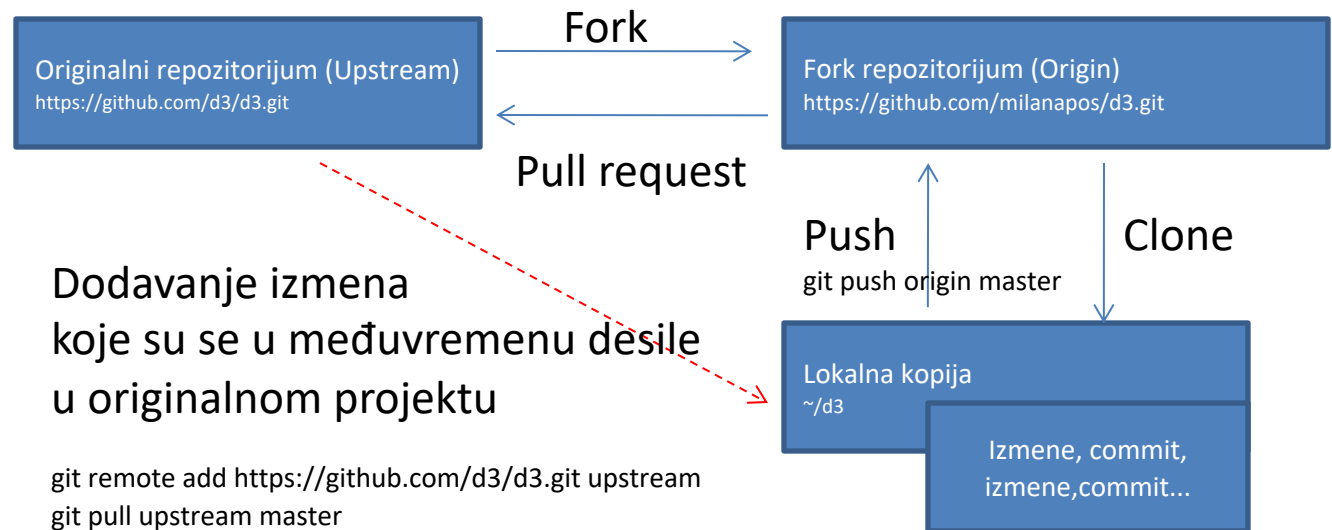
- [R](#)



Open source: Kako se priključiti projektu?

- Zavisi od projekta, ali neki tipični workflow se svodi na sledeće korake:

1. fork
2. clone
3. push
4. pull



Literatura

- <http://pcottle.github.io/learnGitBranching/>
 - Fokus na ovom kursu su samo oblasti
 - Ramping Up
 - Moving Work around
- <https://git-scm.com/doc>
 - Kompletna dokumentacija za `git`