

# Git sistem za kontrolu verzija

- 1 Uvod
- 2 Osnovne git komande
- 3 Rad sa granama
- 4 Rad sa udaljenim repozitorijumom
- 5 Github

# Sistemi za kontrolu verzija

- Na projektima obično radi veći broj ljudi koji paralelno rade
- Javlja se potreba za alatom koji će omogućiti da svaki programer može nezavisno da doprinese projektu
- Sistemi za kontrolu verzija (Version Control System - VCS) vode brigu o promenama datoteka u projektu, uključujući i izvorni kod
- Promena datoteke == nova verzija projekta

# Šta je Git?

- Git je distribuirani alat za kontrolu verzija
  - Kod distribuiranog sistema svaki programer na svom računaru (u lokalu) ima svoju verziju projekta
  - Programeri zatim međusobno razmenjuju verzije
- Git je verovatno najpopularniji sistem ovog tipa i predstavlja de facto standard u softverskoj industriji
- Osnovni radni tok (workflow) je veoma jednostavan, dok su napredne stvari omogućene velikim izborom komandi
- Git ne treba mešati sa github-om i gitlab-om jer su to servisi bazirani na gitu
- Drugi popularni VCS su SVN, Mercurial, Bazaar i sl

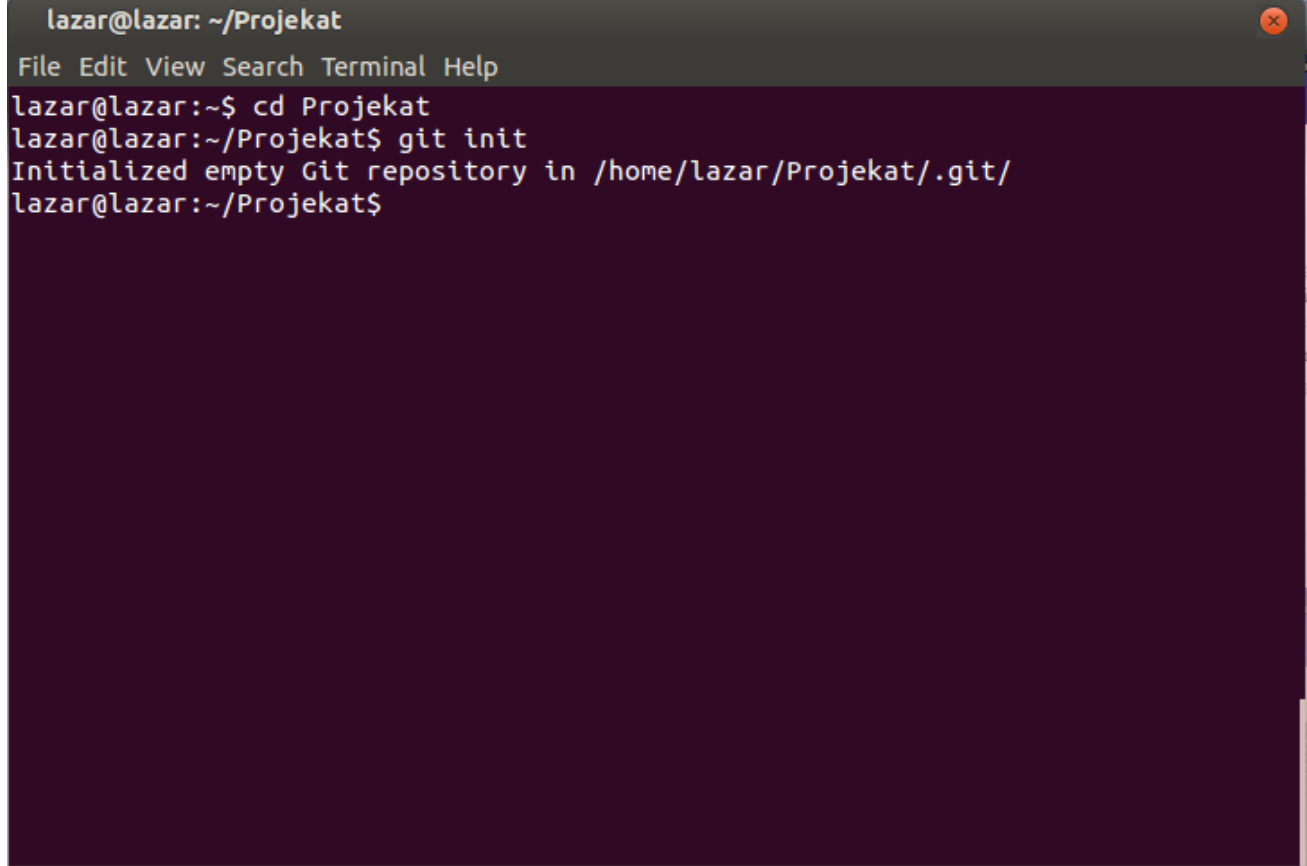
# Kako radi git?

- Git prati sadržaj nekih datoteka
- Git ume da prepozna tačno koji se deo neke datoteke promenio, tako da može vrlo lako da poredi dve verzije neke datoteke
- Ovo omogućava laku razmenu promena između programera unutar tima
- NAPOMENA: Primeri ilustruju upotrebu gita sa običnim tekstualnim datotekama. Kako su Java klase takođe tekstualne datoteke, sve isto važi i za njih

- Git repozitorijum je direktorijum sa projekatom koji treba verzionirati
- Repozitorijum može:
  - Lokalni (local) ako se nalazi samo na vašoj mašini
  - Udaljeni (remote) ako se nalazi na drugoj mašini i često je potrebna internet konekcija da bi mu se pristupilo
- Svaki programer ima svoju verziju repozitorijuma koji se zove radno stablo (working tree)

- Git repozitorijum se kreira tako što se u komandnoj liniji prvo pozicionira na željeni direktorijum
  - `cd putanja_do_direktorijuma`
- Zatim treba pozvati komandu `git init`
- Git će napraviti skriveni direktorijum `.git` koji ne treba dirati

# git init

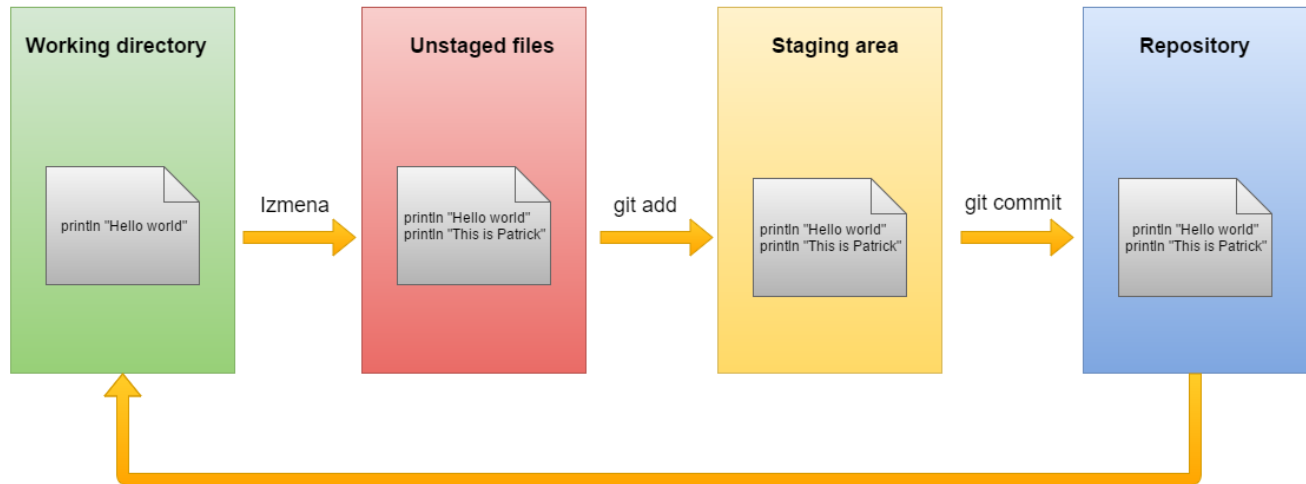


A terminal window titled "lazar@lazar: ~/Projekat" with a menu bar (File, Edit, View, Search, Terminal, Help) and a close button. The terminal shows the following commands and output:

```
lazar@lazar:~$ cd Projekat
lazar@lazar:~/Projekat$ git init
Initialized empty Git repository in /home/lazar/Projekat/.git/
lazar@lazar:~/Projekat$
```



# git workflow



- Promene se trajno beleže kao commit-ovi
- Jedan commit sadrži promene nad datotekama i predstavlja jednu verziju projekta
- Jedinstveno se identifikuje dugačkim nizom karaktera koji se zove hash kod
- Commitovanje se vrši komandom `git commit`
- Commitovi imaju poruku kojom programer opisuje šta je promenio
- Kada se pozove `git commit` otvoriće se datoteka u nekom editoru gde se piše poruka commita
  - Preporučuje se da se u prvoj liniji poruke ukratko opiše commit
  - U ostalim linijama može ići detaljniji opis
  - Sve iza znaka `#` je komentar i ne ulazi u poruku

- Kada se neka datoteka promeni, te promene se mogu "spremiti" za commit komandom `git add`
- Datoteke sa promenama se onda smeštaju u tzv. "staging" ili "index" zonu
- Sve datoteke u ovoj zoni će se uključiti u commit kada se pozove komanda `git commit`

# git add primer



```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
lazar@lazar:~/Projekat$ git add hola.txt
```

# git add primer

```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
lazar@lazar:~/Projekat$ git commit hola.txt
```

```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
Dodat pozdrav na spanskom
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# Explicit paths specified without -i nor -o; assuming --only paths...
# On branch master
# Changes to be committed:
#   new file:   hola.txt
#
# Changes not staged for commit:
#   modified:   hello.txt
#
# Untracked files:
#   zdravo.txt
#
-- INSERT --
1,26 All
```

- Komanda `git status` prikazuje trenutno stanje repozitorijuma
- Datoteke spremne za dodavanje (one koje su addovane) su napisane zelenim slovima ispod rečenice "Changes to be committed:")
- Promenjene datoteke koje nisu dodate su napisane crvenim slovima ispod rečenice "Changes not staged for commit:"
- Nove datoteke koje još uvek nisu dodate u repozitorijum su napisane crvenim slovima ispod rečenice "Untracked files:"
- Vrlo je korisno videti koji su fajlovi promenjeni pre commita

# git status

```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
lazar@lazar:~/Projekat$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   hola.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

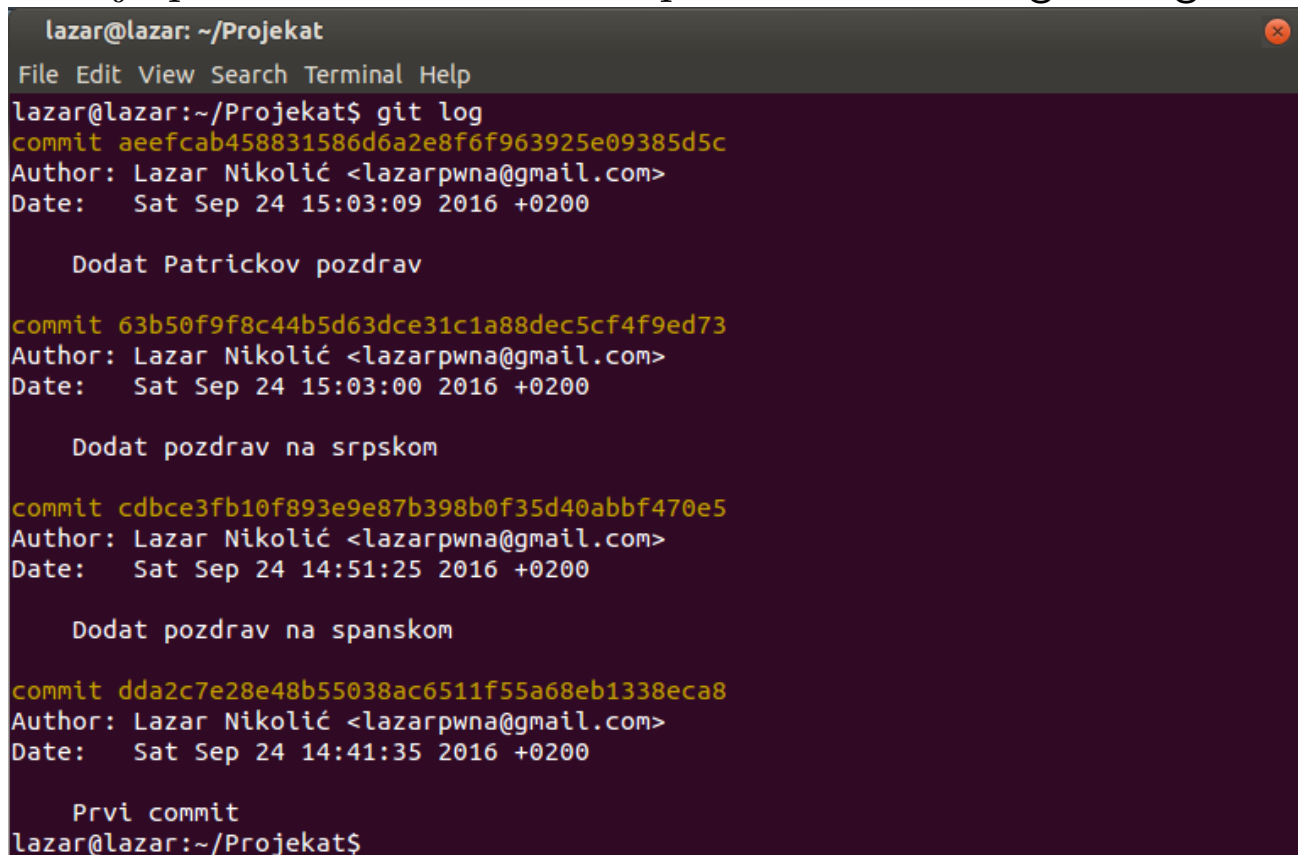
    modified:   hello.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    zdravo.txt

lazar@lazar:~/Projekat$
```

Istorija promena se može videti pomoću komande `git log`



```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
lazar@lazar:~/Projekat$ git log
commit aeefcab458831586d6a2e8f6f963925e09385d5c
Author: Lazar Nikolić <lazarpwna@gmail.com>
Date: Sat Sep 24 15:03:09 2016 +0200

    Dodat Patrickov pozdrav

commit 63b50f9f8c44b5d63dce31c1a88dec5cf4f9ed73
Author: Lazar Nikolić <lazarpwna@gmail.com>
Date: Sat Sep 24 15:03:00 2016 +0200

    Dodat pozdrav na srpskom

commit cdbce3fb10f893e9e87b398b0f35d40abbf470e5
Author: Lazar Nikolić <lazarpwna@gmail.com>
Date: Sat Sep 24 14:51:25 2016 +0200

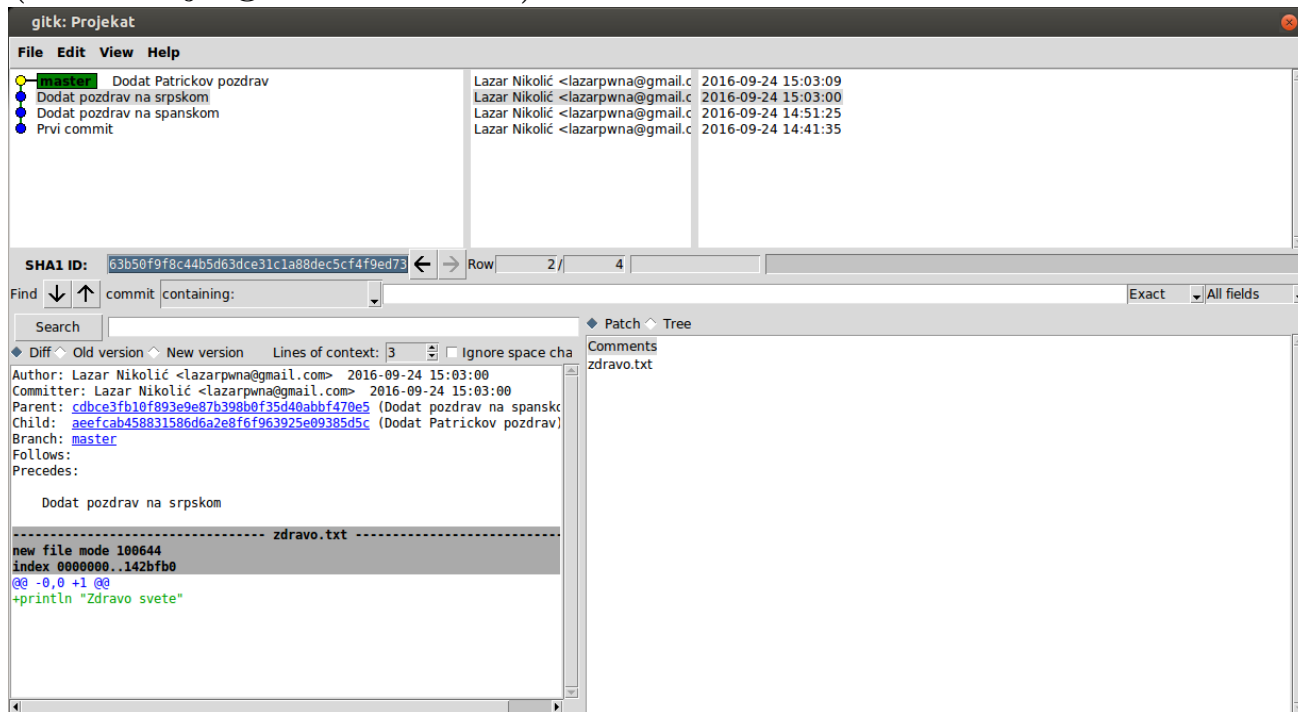
    Dodat pozdrav na spanskome

commit dda2c7e28e48b55038ac6511f55a68eb1338eca8
Author: Lazar Nikolić <lazarpwna@gmail.com>
Date: Sat Sep 24 14:41:35 2016 +0200

    Prvi commit
lazar@lazar:~/Projekat$
```



Istorija promena se može grafički prikazati pomoću komande `gitk` (ukoliko je `gitk` instaliran)



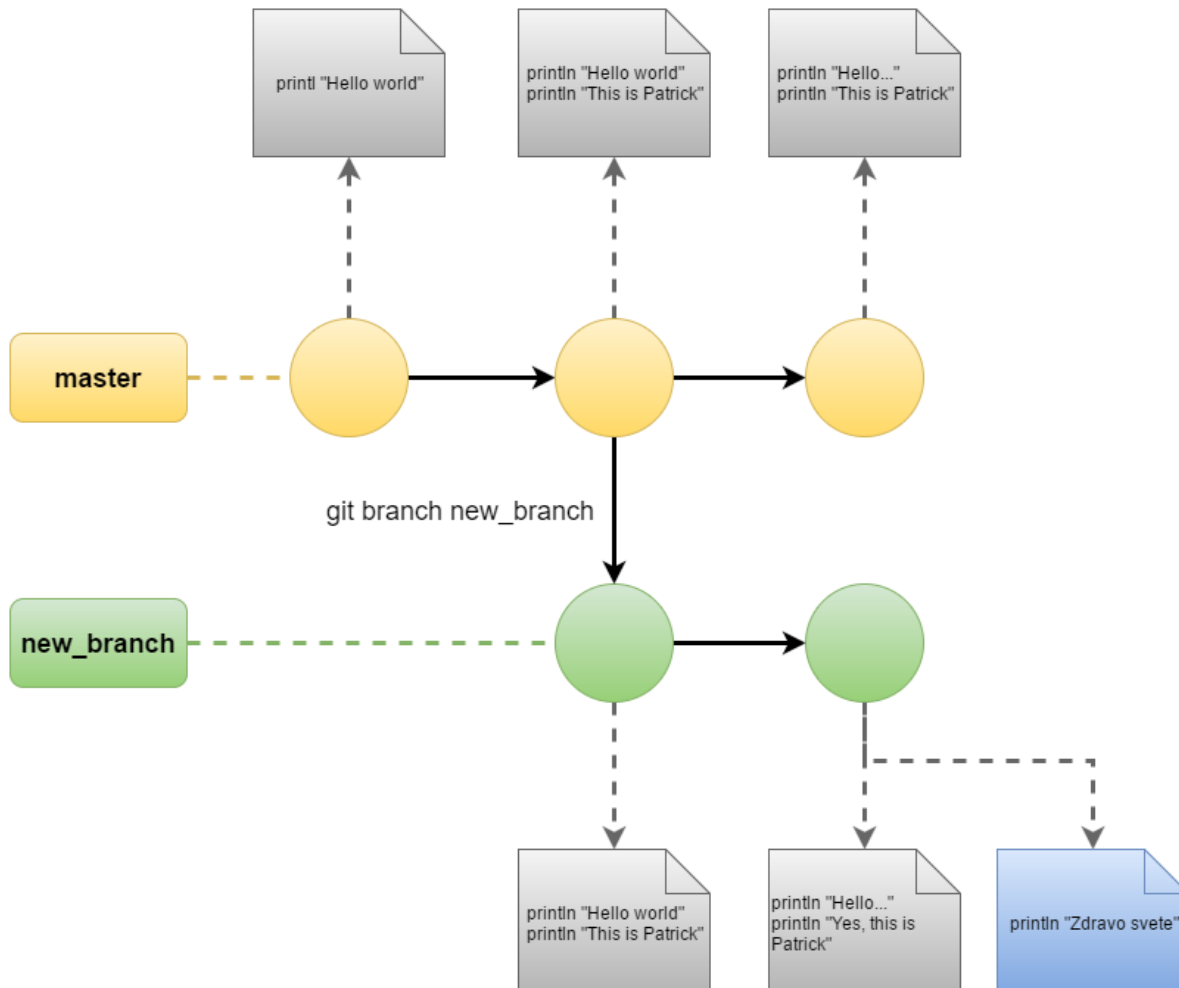
# Ignorisanje datoteka

- Git nudi mogućnost ignorisanja datoteka koje ne želimo u repozitorijumu
- One su još uvek tu, ali se ne mogu commitovati
- Ovo je potrebno jer je git dobar za rad sa tekstom, ali ne može da poredi dve binarne datoteke kao što su slike ili programske biblioteke

- U .gitignore se navodi koje datoteke git ne treba da prati
- Najčešće sadrži binarne datoteke (slike i ostale multimedije), programske biblioteke, generisane stvari (npr. dokumentacija) i stvari vezane za okruženje koje koristite (.idea direktorijum)
- Ignorisanje se vrši na sledeći način:
  - Ignorisanje pojedinačnog direktorijuma: /ime\_direktorijuma (npr. /bin)
  - Ignorisanje pojedinačne datoteke (npr. slika.png)
  - Ignorisanje svih datoteka neke vrste (npr. \*.png ignoriše sve slike formata png)

- Grane u gitu su alternativni tokovi razvoja
- Svaki git repozitorijum ima jednu podrazumevanu glavnu granu: master
- Svaka grana ima nezavisno radno stablo (working tree)
  - Ista datoteka može biti različita na dve različite grane
  - Jedna datoteka može da postoji na jednoj, a da ne postoji na drugoj grani
- Kreiranje nove grane u gitu se vrši komandom `git branch <ime_grane>`
  - Grana na kojoj se nalazimo je označena zvezdicom (\*)
- Prikaz svih grana se vrši komandom `git branch`

# git branch



# git branch

```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
lazar@lazar:~/Projekat$ git branch druga_grana
lazar@lazar:~/Projekat$ git branch
  druga_grana
* master
lazar@lazar:~/Projekat$
```

- Komanda `git checkout <ime_grane>` služi za prebacivanje na neku granu
- Takođe je moguće vraćanje na staru verziju gita se obavlja komandom `git checkout <hash_kod_commita>`
- Git onda ulazi u DETACHED stanje
- U ovom stanju nije moguće dodavati nove commitove, ali je moguće kreirati novu granu pa krenuti od nje
- Vraćanje na aktuelnu verziju se vrši komandom `git checkout <ime_trenutne_grane>` (npr. `git checkout master`)

# git checkout na granu

```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
lazar@lazar:~/Projekat$ git checkout druga_grana
Switched to branch 'druga_grana'
lazar@lazar:~/Projekat$ git branch
* druga_grana
  master
lazar@lazar:~/Projekat$
```



# git checkout na commit

```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
lazar@lazar:~/Projekat$ git checkout cdbce3fb10f893e9e87b398b0f35d40abbf470e5
Note: checking out 'cdbce3fb10f893e9e87b398b0f35d40abbf470e5'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

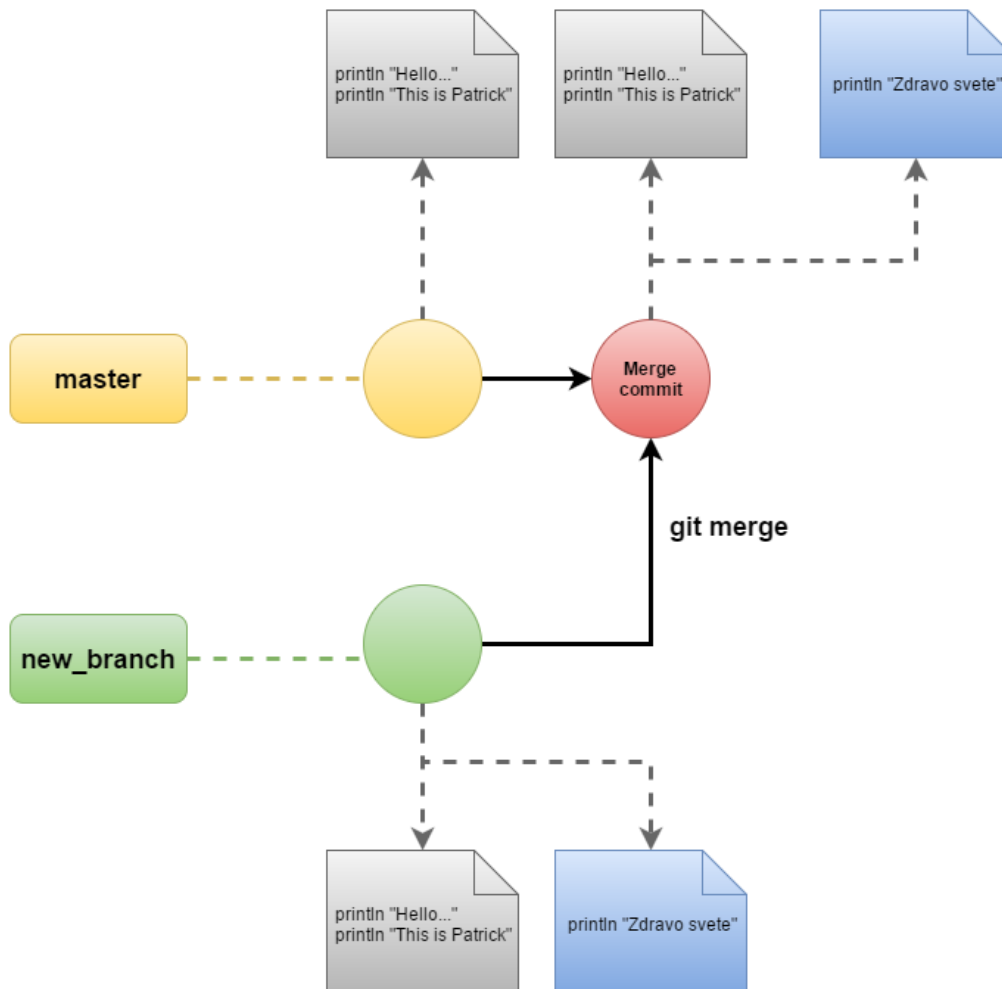
    git checkout -b new_branch_name

HEAD is now at cdbce3f... Dodao pozdrav na spanskom
lazar@lazar:~/Projekat$
```

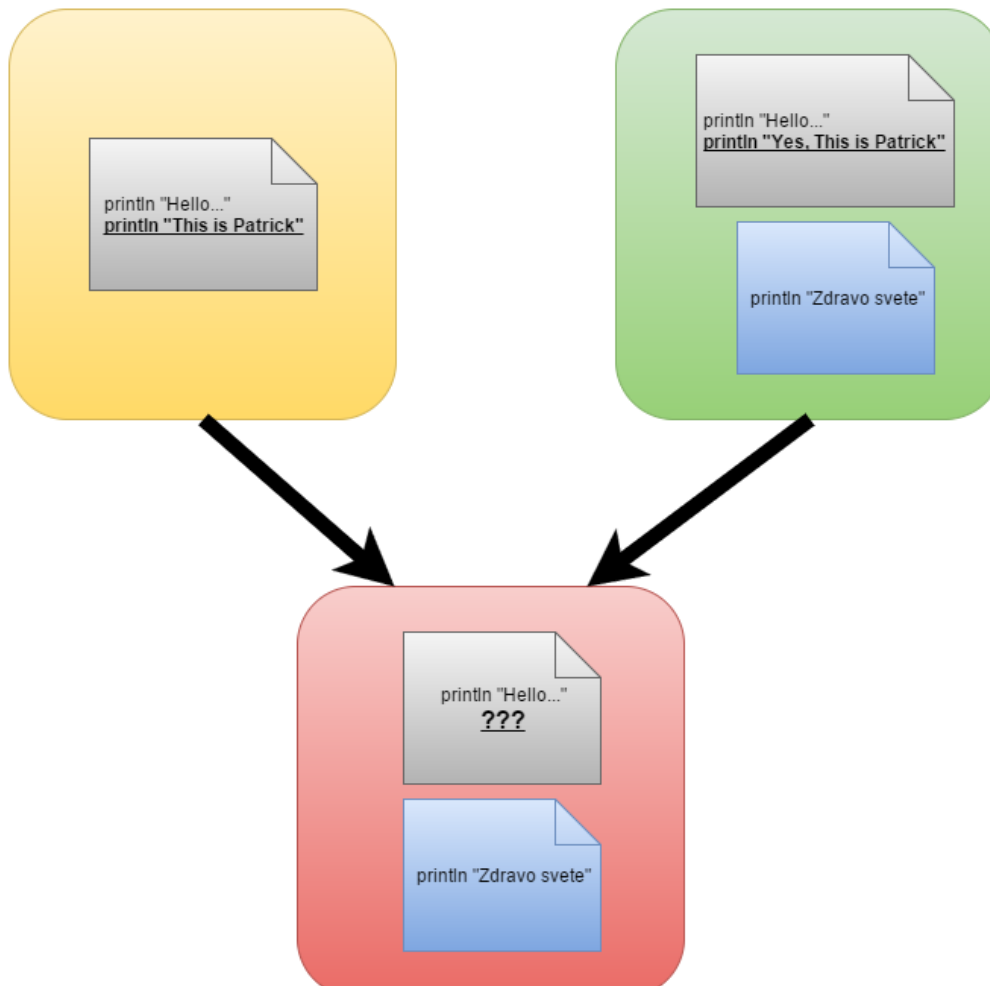
# Spajanje grana

- Eventualno dve grane se mogu spojiti
- Git će pokupiti commitove sa obe grane i pokušaće da spoji izmene
- Pravi se specijalni Merge commit koji opisuje kako je spajanje prošlo
- Spajanje dve grane se vrši komandom  
`git checkout merge <ime_grane>` (spoji ovu granu na trenutnu)

# Spajanje dve grane



# Konflikt



- Konflikt nastaje kada se u jednoj datoteci promene iste linije koda
- Git u tom slučaju ne zna kako da spoji datoteke sa dve grane, pa prijavljuje konflikt
- Rešavanje konflikta:
  - Ručno promeniti sve datoteke koje su u konfliktu
  - `git add` za svaku datoteku u konfliktu
  - `git commit`
  - Može se zadržati podrazumevana poruka

# Konflikt

```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
lazar@lazar:~/Projekat$ git merge druga_grana
Auto-merging hello.txt
CONFLICT (content): Merge conflict in hello.txt
Automatic merge failed; fix conflicts and then commit the result.
lazar@lazar:~/Projekat$
```

```
lazar@lazar: ~/Projekat
File Edit View Search Terminal Help
println "Hello..."
<<<<<< HEAD
println "...This is Patrick"
=====
println "... Yes, this is Patrick"
>>>>>> druga_grana

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

- Konflikt nastaje kada se u jednoj datoteci promene iste linije koda
- Git u tom slučaju ne zna kako da spoji datoteke sa dve grane, pa prijavljuje konflikt
- Rešavanje konflikta:
  - Ručno promeniti sve datoteke koje su u konfliktu
  - `git add` za svaku datoteku u konfliktu
  - `git commit`
  - Može se zadržati podrazumevana poruka

# Rad sa udaljenim repozitorijumom (1)

- Kreiranje novog udaljenog repozitorijuma se obavlja na sledeći način:
  - Inicijalizovati repozitorijum sa `git init`
  - Dodati neki fajl i komitovati ga
  - Pozvati komandu  
`git remote add origin <adresa_repozitorijuma>`
  - Pozvati komandu `git push -u origin master`
- Rad sa postojećim udaljenim repozitorijumom počinje komandom `git clone <adresa_repozitorijuma>`
- Svi commitovi koje uradimo se nalaze u lokalnom repozitorijumu
- Da bismo objavili promene svima koristimo komandu `git push`
- Da bismo dobavili tuđe promene, koristimo komandu `git pull`
  - `git pull` može izazvati konflikte koje treba razrešiti na isti način kao `git merge`



# Rad sa udaljenim repozitorijumom (2)

- Pre pushovanja se moraju pullovati izmene kako bismo ostali sinhronizovani
- Tipičan git workflow:
  - Izmena/dodavanje datoteke
  - `git add`
  - `git commit`
  - `git pull`
  - Razrešavanje konflikta, ako postoje
  - `git push`

- Github je server za git
- Izuzetno popularan, sav kod je otvoren (može se platiti da bude privatan), može poslužiti kao CV za programera
- Omogućava kreiranje udaljenih repozitorijuma, projekata i izdavanje issue-a za članove tima

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



/

Repository name

Great repository names are short and memorable. Need inspiration? How about **special-potato**.

Description (optional)



**Public**

Anyone can see this repository. You choose who can commit.



**Private**

You choose who can see and commit to this repository.



**Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▼

Add a license: **None** ▼




Create repository

- Github issue je zadatak, predlog ili prijava buga za tim na projektu
- Može ga otvoriti bilo ko i dodeliti bilo kome
- Može imati tagove koji označavaju prirodu issue-a (feature, bug, question itd)
- Svaki issue ima redni broj iza # (npr. #2, #5, #10)
- Issue se može zatvoriti kada je razrešen
- Ovo se automatski vrši ako se u commit poruci napiše fixes #broj, closes #broj, resolves #broj i sl

# Github issue

[Code](#) [Issues 2](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Pulse](#) [Graphs](#) [Settings](#)




Write

Preview

AA ▾ B i “ ” < > ↺ ⋮ ≡ ≡ ≡ ↶ @ 📌

Leave a comment

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

 Styling with Markdown is supported

Submit new issue

Labels

None yet

Milestone

No milestone

Assignees

No one—assign yourself