

Da vidimo sada kako se koristi komanda **git ignore** (kako da ne prikazujemo neke stvari)

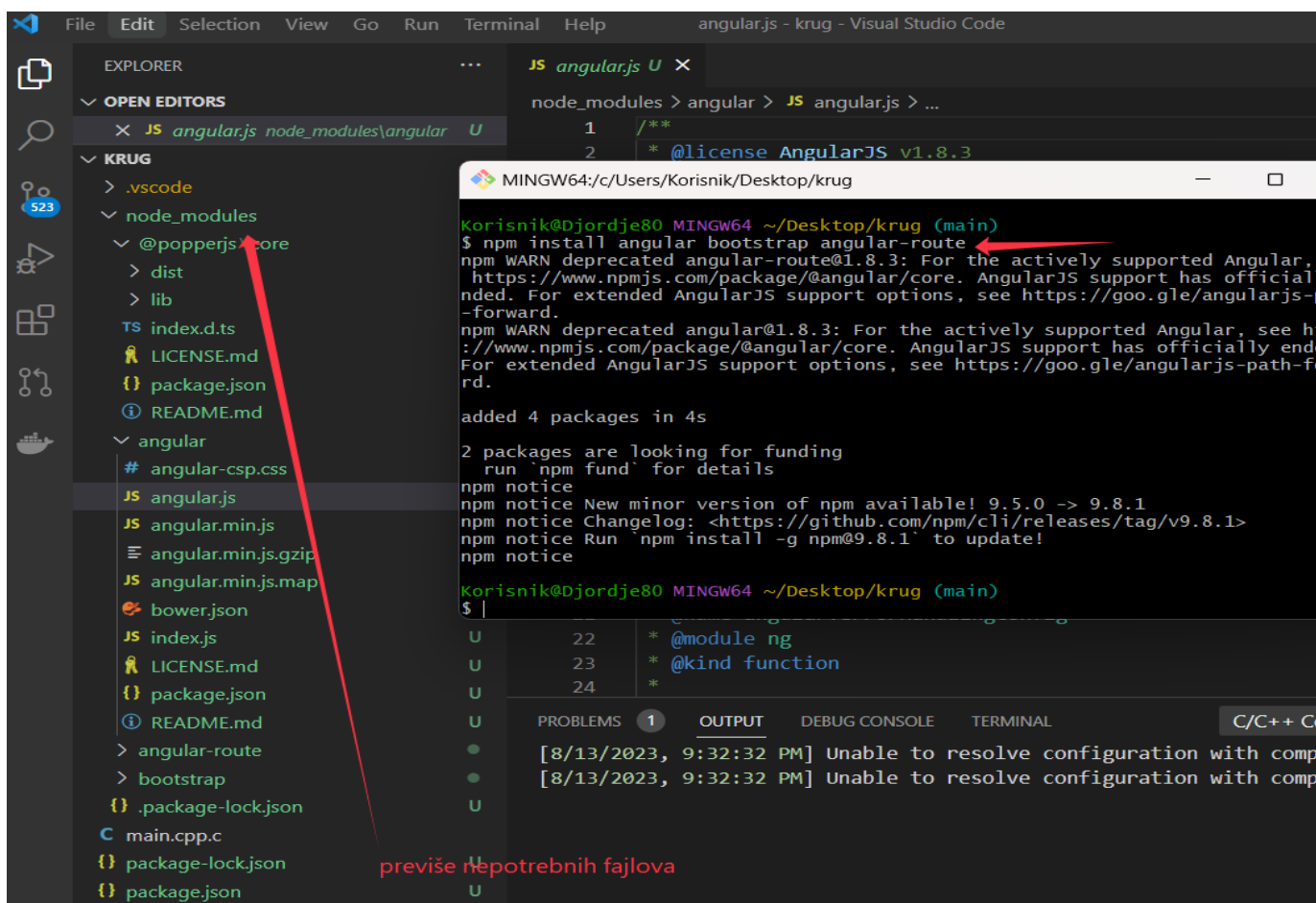
Kopirao, klonirao sam reposetorijum sa programom krug.

Instalirajmo sada npr node module sa npm

Tj kucamo.

npm install angular bootstrap angular-route

Postoje fajlovi, folderi koje je nepotrebno deliti .

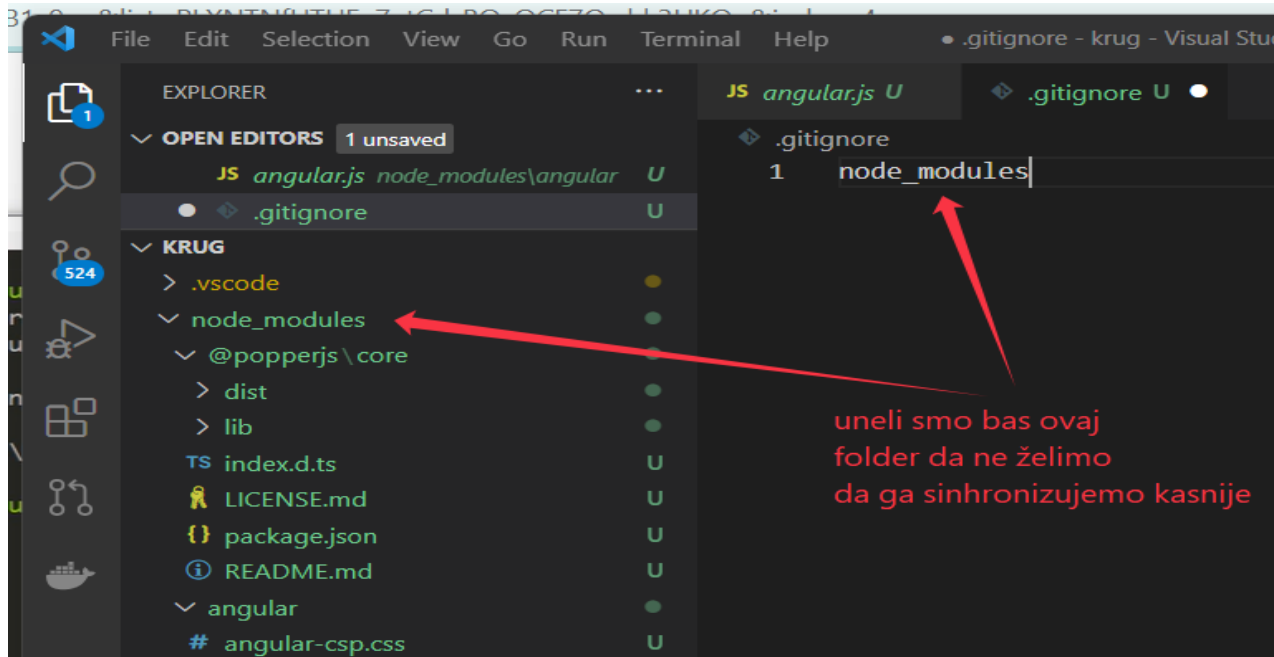
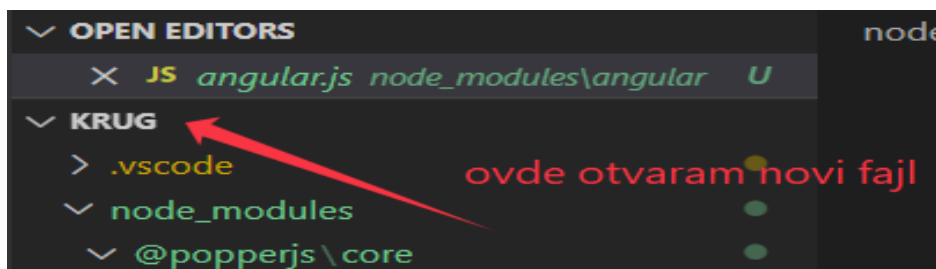


Instalirao sam sada neke node module čisto da vidite kada stavim opciju **git ignore** na neke nepotrebne stvari.

Ima kao što vidimo previše nebitnih fajlova tako da ćemo kucati u svom glavnom folderu **novi fajl**

.gitignore (tu ćemo upisati šta da sam git softver ignoriše kada treba da sinhronizuje naše fajlove)

Mi ćemo staviti baš ovaj folder koji nam je nepotreban a učitali smo ga demonstrativno **node_modules**



Vratimo se sada u naš terminal i pogledajmo **git status** da vidimo šta on pokazuje.

```
korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  .gitignore
  .vscode/
  package-lock.json
  package.json

nothing added to commit but untracked files present (use "git add" to track)

korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
```

ono sto nas zanima

Imamo napravljeni gitignore fajl koji želimo da push-ujemo i idemo sa komandom `git add .` (. -Označava da sve dodajemo izmene na github)

Onda ide za potvrdu komanda `git commit -m " Samo test ..."`

Pa da onda to sinhronizujemo sa `git push`

```
Korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ git add .
warning: in the working copy of '.vscode/c_cpp_properties.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.vscode/launch.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.vscode/settings.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it

Korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ git commit -m " samo proba "
[main 6d9b388] samo proba
6 files changed, 162 insertions(+)
create mode 100644 .gitignore
create mode 100644 .vscode/c_cpp_properties.json
create mode 100644 .vscode/launch.json
create mode 100644 .vscode/settings.json
create mode 100644 package-lock.json
create mode 100644 package.json

Korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 2.34 KiB | 599.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/djoemladenovic/krug.git
fabddc8..6d9b388 main -> main

Korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ |
```

Idemo sada na naš GitHub i pogledajmo naš folder krug da vidimo da li se i ikako izvršila sinhronizacija.

main 1 branch 0 tags Go to

| | | |
|----------------------------|-------------------|-------------------|
| djolemladenovic samo proba | | |
| 📁 | .vscode | samo proba |
| 📄 | .gitignore | samo proba |
| 📄 | README.md | Initial commit |
| 📄 | main.cpp.c | Update main.cpp.c |
| 📄 | package-lock.json | samo proba |
| 📄 | package.json | samo proba |

vidimo naš fajl koji smo pravili

krug / .gitignore

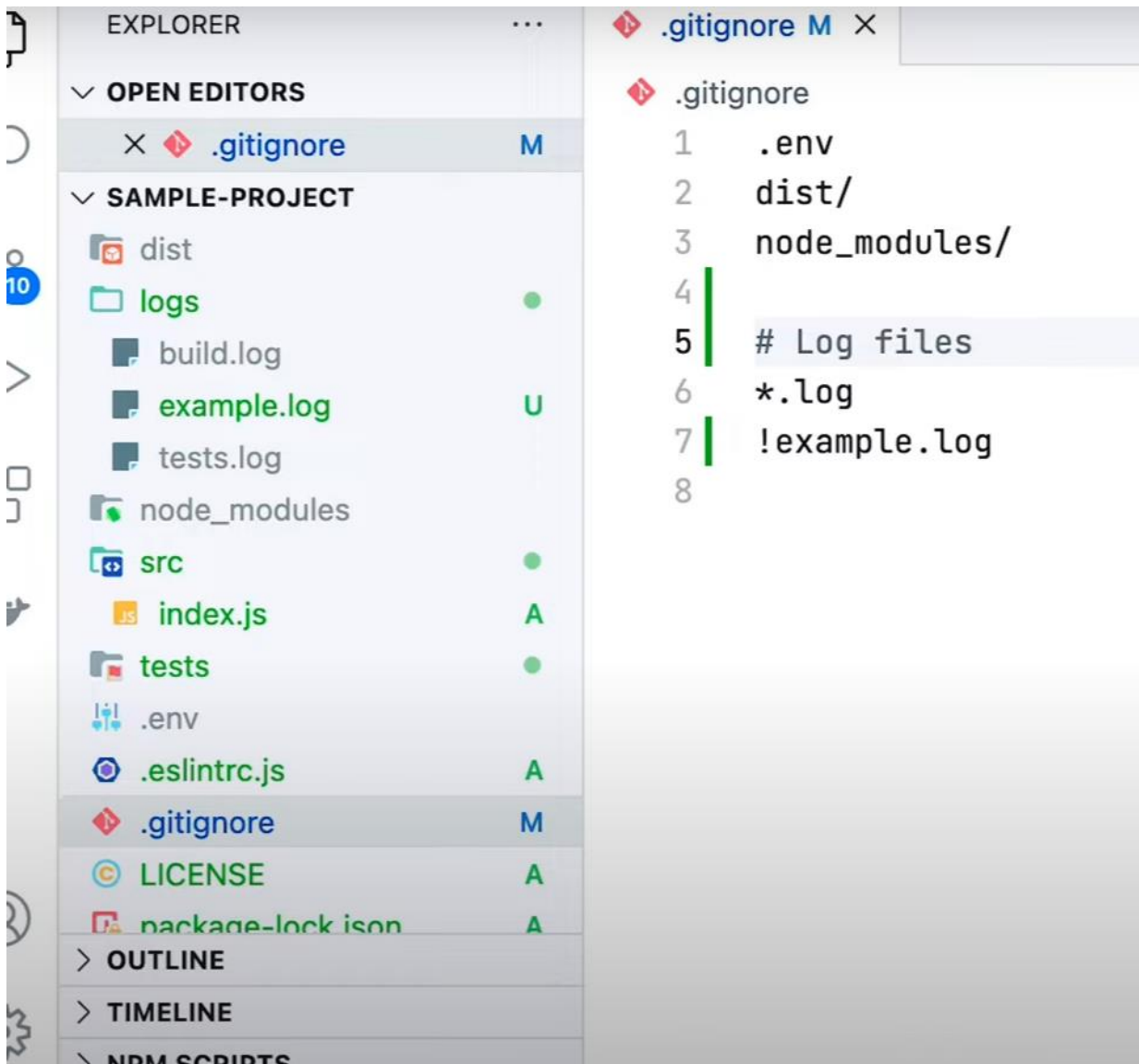
djolemladenovic samo proba

Code Blame 1 lines (1 loc) · 12 Bytes kao što vidimo u tom fajlu je samo ovo

```
1 node_modules
```

Kao što vidimo u git ignore fajlu je samo jedna stvar a ne brdo fajlova koje bi korisnici morali svaki put da kopiraju kod sebe .

Ovde smo videli kako da napravimo fajl kada želimo da se nešto ne uključimo u naš projekat.



Ako želimo da ipak neki log fajl izuzmemo , kucaćemo **!example.log**

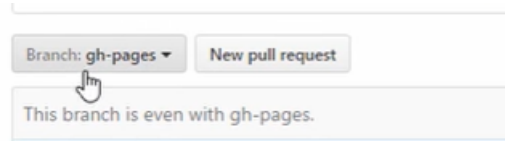
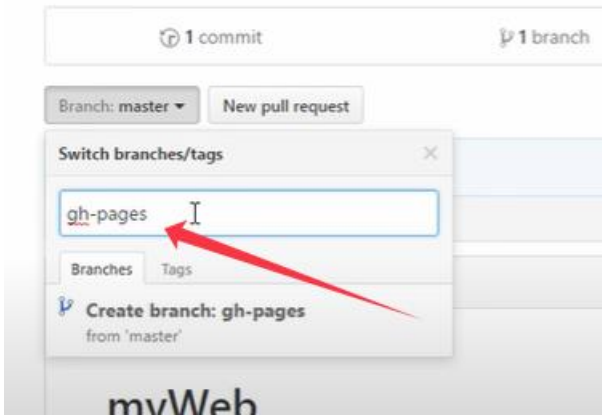
Ako kucamo ***.log** to se podrazumeva na sve fajlove koji se u logs

Da vidimo šta je GitHub Pages

To nije ništa drugo nego besplatan hosting za naš web sajt.

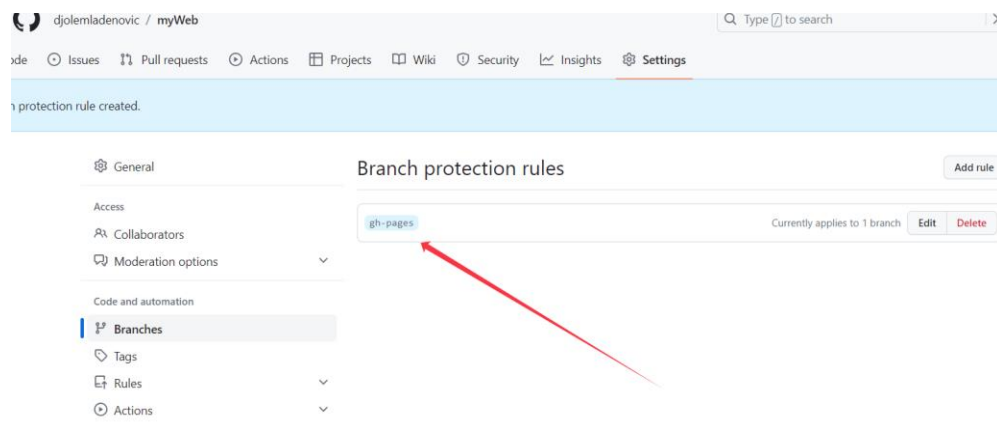
Krećemo od pravljenja new repository na GitHubu. Ukucajte kako želite da se zove novi repozitorijum.

Mi smo tu na našem **master Branch-u** , tu ćemo ukucati **gh-pages**



I kreiraćemo taj novi Branch

Tu i dalje postoji master brench (main) i on je da kažem i dalje glavni brench e to moramo promeiti u setings-u.



Kada ponovo odemo na naš dokument myWeb i u setings-u odemo na opciju Pages možemo videti da je sada prikazana putanja do naše veb strane.

GitHub repository settings for **myWeb**. The **Pages** section is highlighted in the left sidebar. The main content shows the site URL <https://djolemladenovic.github.io/myWeb/> and the **Build and deployment** section with **gh-pages** branch selected. Red arrows point to the **Settings** link, the **Pages** sidebar item, and the URL field. A red note says "ovde možemo videti naš sajt".

Sada naravno nemam ništa još uvek zato što mi je folder prazan ali hajde da stavimo neki fajl i da napišemo neki kod.

Nazovimo ga **index.html** (naravno prva stranica koja će se otvoriti)

GitHub repository code view for **myWeb** showing the **index.html** file. The file name **index.html** is highlighted in the path bar. The **Edit** button is visible. A red arrow points to the **index.html** text, and another red arrow points to the **Enter file contents here** text in the editor area. A red note says "kod ćemo kucati u VSCodeu pa ćemo ga ovde preneti".

Uneo sam sada neki program pisan u html-u i sacuvacu izmene tj. **commit changes**

myWeb / index.html in main

Cancel changes

Commit changes...

Edit

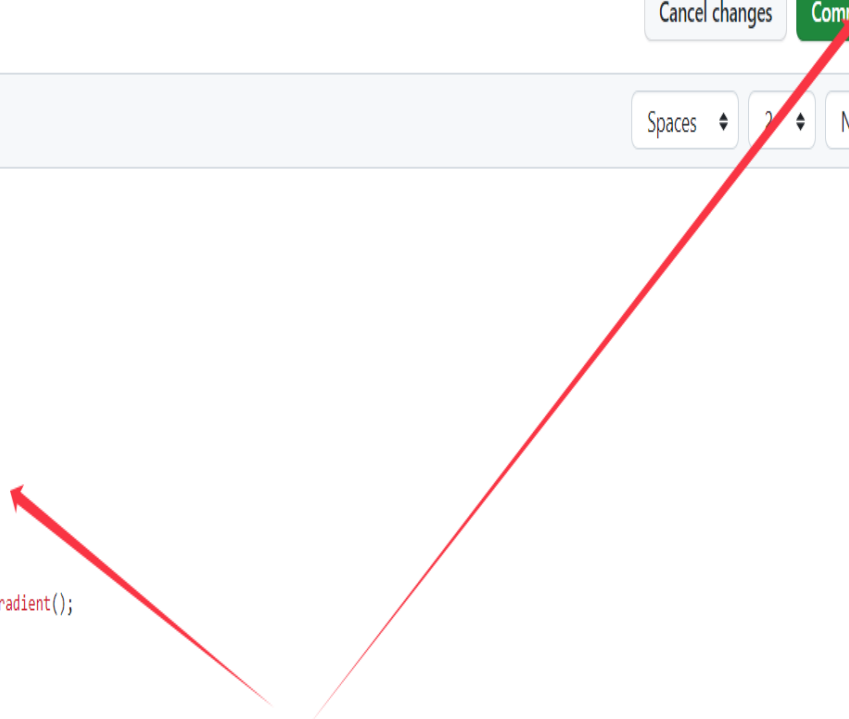
Preview

Spaces

2

No wrap

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title> myWeb</title>
6     <style media="screen">
7       h1 {
8         color:blueviolet;
9         font-size: 4em;
10        font-size: larger;
11        mask-image: -moz-linear-gradient();
12      }
13    </style>
14  </html>
15  <h1> Moj prvi sajt</h1>
16 </body>
17
```



General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://djolemladenovic.github.io/myWeb/>
Last deployed by [djolemladenovic](#) 46 minutes ago

Visit site



Build and deployment

Source

Deploy from a branch

Branch

Your GitHub Pages site is currently being built from the gh-pages branch. [Learn more about configuring the publishing source for your site.](#)

gh-pages

/(root)

Save

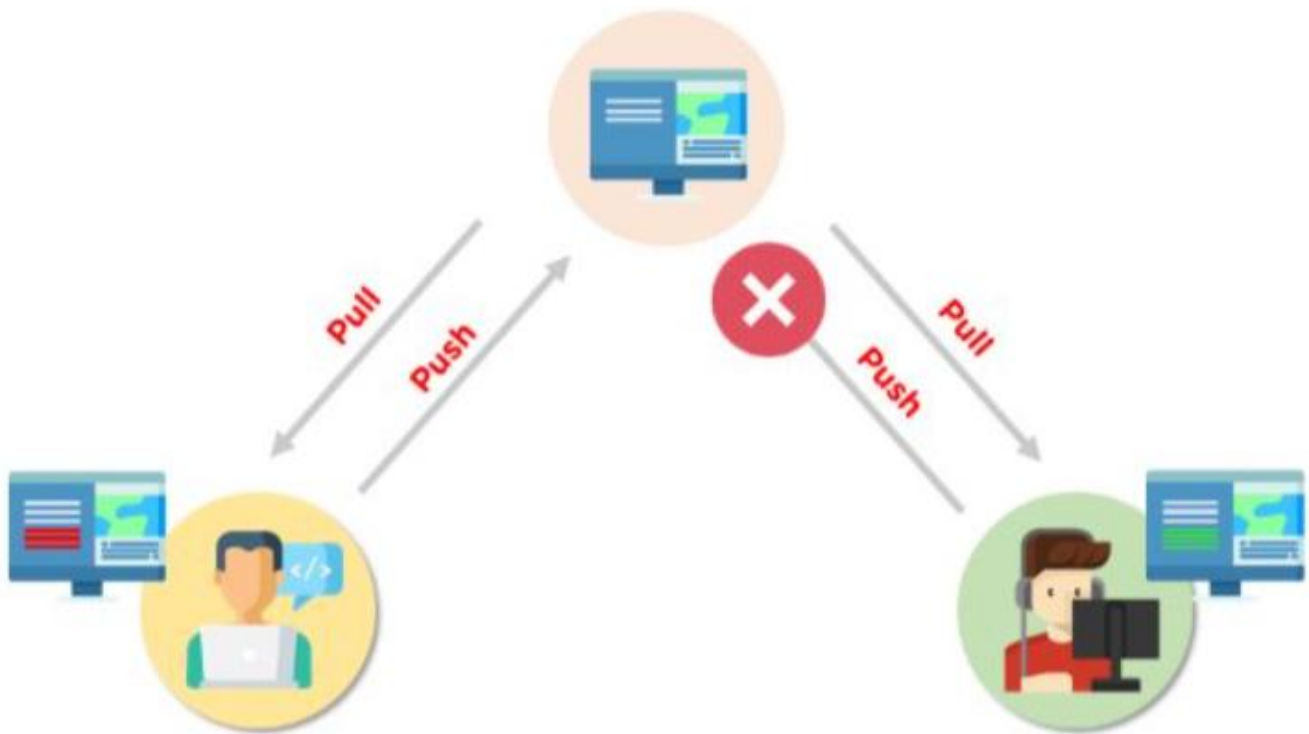
Learn how to [add a Jekyll theme](#) to your site.

Sada možemo otići na naš sajt

What is a Git Merge Conflict? (Šta je neusaglašenost pri objedinjavanju)

Neusaglašenost pri objedinjavanju git je događaj koji se dešava kada Git nije u stanju da automatski reši razlike u kodu između dva komita. Git može automatski da objedini promene samo ako su komiti na različitim linijama ili granama.

Sledi primer kako funkcioniše neusaglašenost git objedinjavanja:



Pretpostavimo da postoje dva programera: Developer A i Developer B. Obojica izvlače isti kodni fajl iz udaljenog skladišta i pokušavaju da naprave razne amandmane u tom dosijeu. Nakon što izvrši izmene, Developer A gura datoteku nazad u udaljeno skladište iz svog lokalnog skladišta. Sada, kada Developer B pokuša da pritisne tu datoteku nakon što izvrši promene sa svog kraja, on to nije u mogućnosti, jer je datoteka već promenjena u udaljenom skladištu.

Da bi sprečili takve konflikte, programeri rade u **odvojenim izolovanim granama**. Komanda za objedinjavanje Git kombinuje zasebne grane i rešava sva neusaglašena uređivanja.

Sada kada smo prošli kroz osnove sukoba Git mergea, pogledajmo sledeće različite vrste konflikata.

Kako otkloniti neusaglašenosti objedinjavanja u Gitu?

Postoji nekoliko koraka koji bi mogli da smanje korake potrebne za rešavanje neusaglašenosti objedinjavanja u Git-u.

1. korak: Najlakši način za otklanjanje neusaglašene datoteke je da je otvorite i izvršite sve neophodne promene.

2. korak: Nakon uređivanja datoteke, možemo da koristimo git dodavanje komande za fazu novog objedinjenog sadržaja.

Korak 3: Poslednji korak je stvaranje novog posvećivanja uz pomoć komande git commit.

4. korak: Git će kreirati novo objedinjavanje da bi dovršio objedinjavanje.

Git Commands to Resolve Conflicts

1. git log --merge

The git log --merge komanda pomaže u izradi liste posvećivanja koja uzrokuje neusaglašenost.

2. git diff

The git diff command pomaže u identifikovanju razlika između depoa ili datoteka država.

3. git checkout

The git checkout command koristi se za opozivanje promena izvršenih u datoteci ili za promenu grana.

4. git reset --mixed

The git reset --mixed command koristi se za opoziv promena u radnom direktorijumu i prostoru za pripremu.

5. git merge --abort

The git merge --abort command pomaže u izlasku iz procesa objedinjavanja i povratku u državu pre početka spajanja.

6. git reset

The git reset command koristi se u vreme neusaglašenosti objedinjavanja da bi se neusaglašene datoteke vratile u prvobitno stanje.

Kao primer u svom Git hab nalogu u file1 sam napravio ismene . Takođe sam na istom fajlu na istom mestu u cloniranom fajlu na terminal napravio isto ismene na istom mestu ali različite.

The screenshot shows a GitHub repository interface. At the top, there is a search bar with the text "Type / to search". Below the search bar is a navigation bar with links for "All requests", "Actions", "Projects", "Wiki", "Security", "Insights", and "Settings". The main content area displays the file path "file1 / hrana" with a copy icon. Below this, a commit by user "djmladenovic" is shown with the title "Update hrana". The commit details include "Code", "Blame", and "5 lines (5 loc) · 38 Bytes". The code content is as follows:

```
1   Doručak
2   Užina
3   Ručak
4   Užina
5   Večera
```

```
Terminal Tue 12:05
djordje80@ubuntu: ~/Desktop/fil
File Edit View Search Terminal Help

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:   hrana

no changes added to commit (use "git add" and/or "git commit -a")
djordje80@ubuntu:~/Desktop/file1$ git add hrana
djordje80@ubuntu:~/Desktop/file1$ git commit -m"Promena u redosledu hrane"
main 578b92e] Promena u redosledu hrane
1 file changed, 2 insertions(+)
djordje80@ubuntu:~/Desktop/file1$ git push
to github.com:djmladenovic/file1.git
! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'git@github.com:djmladenovic/file1.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
djordje80@ubuntu:~/Desktop/file1$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
unpacking objects: 100% (3/3), done.
from github.com:djmladenovic/file1
   fc594ac..e26f384  main      -> origin/main
auto-merging hrana
CONFLICT (content): Merge conflict in hrana
automatic merge failed; fix conflicts and then commit the result.
djordje80@ubuntu:~/Desktop/file1$
```

Tada se pojavio git conflit.

```
File Edit View Search Terminal Help
hint: to the same ref. You may want to first integrate the
remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --hel
p' for details.
djordje80@ubuntu:~/Desktop/file1$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused
0
Unpacking objects: 100% (3/3), done.
From github.com:djmladenovic/file1
   fc594ac..e26f384  main      -> origin/main
Auto-merging hrana
CONFLICT (content): Merge conflict in hrana
Automatic merge failed; fix conflicts and then commit the r
esult.
djordje80@ubuntu:~/Desktop/file1$
```

Konflikt je tu , sada se vratimo u VC I pogledamo te koflikte

Videcete koje opcije postoje. Imamo mogucnost da zadržimo prvi, drugi ili oba . Pa I da ih uporedimo.

