



(Versiyon Kontrol Sistemi)

Ender ÇELİK
celikk.ender@gmail.com



Bölüm 1: Git Nedir?

- Git, açık kaynak kodlu bir versiyon kontrol sistemidir.
- Linus Torvalds tarafından 2005 yılında geliştirilmiştir.
- Mercurial, SVN, Bazaar vb... gibi alternatifleri vardır.



Bölüm 2:

Git Terimleri & Başlıca Git Komutları

repository:

- Proje dosyalarının ve bu dosyalar üzerinde yapılan değişikliklerin saklandığı, takip edildiği, yönetildiği bir dizini/depoyu ifade eder.
- Lokal bir bilgisayar veya uzak bir sunucu üzerinde bulunabilir. Genellikle projenin ana depoları uzak bir sunucuda (GitHub, GitLab, Bitbucket gibi platformlarda) barındırılır.
- Kısaca “repo” olarak ifade edilir.

.gitignore:

- **“.gitignore”** dosyası, bir Git deposundaki belirli dosyaların veya dizinlerin izlenmemesini (ignore edilmesini) sağlamak için kullanılan bir yapılandırma dosyasıdır.
- Genellikle derleme çıktılarının, kullanıcı özel ayar dosyalarının ve benzeri geçici veya özel dosyaların depo tarafından izlenmemesi için kullanılır.



.gitignore:

- NodeJs'de modülleri içeren “/node_modules” dizini ya da Qt'deki “.pro.user” dosyası gibi birimler .gitignore dosyasının içeriğini oluşturur.

```
3 build/
4
5 pubspec.lock
6
7 doc/api/
8
9 C:\vsCodeRepo\yuruyus_app\android\app\google-services.json
10 C:\vsCodeRepo\yuruyus_app\android\app\src\main\AndroidManifest.xml
11
12 # dotenv environment variables file
13 .env*
14
15 *.js.deps
16 *.js.map
```

branch:

- Projenin “birbirinden izole edilmiş” kopyaları halindeki her bir geliştirme yolunu temsil eder.
- Projedeki farklı özellikleri veya geliştirmeleri aynı anda yönetmek için kullanılırlar.
- Projenin ana branch’i “master” veya “main” olarak adlandırılır.

branch:

- Projeye yeni özellik ekleme işlemi için **feature branch**, projedeki bir hatayı düzeltmek için **bugfix branch** oluşturabilir; güvenli bir şekilde geliştirme ve testleri yapabilirsiniz.
- Planladığınız duruma getirdiğiniz branch'i **merge** komutu ile master branch'inize entegre edebilirsiniz.

branch:



commit:

- Yapılan değişikliklerin yerelde kalıcı olarak kaydedildiği yapı. (komut)
- 40 karakterlik bir alfanümerik dizeden oluşan “Commit kimliği” veya “Commit Hash” olarak adlandırılan unique ID ile tanımlanırlar. (reset, revert komutlar)

```
C:\Users\Ender\Desktop\vcs_presentation>git log
commit 1ce1481a955ee95ef133fbb6f90ff256c865bf33 (HEAD -> main)
Author: Ender ÇELİK <celikk.ender@gmail.com>
Date: Sun Dec 17 19:36:44 2023 +0300

    second commit

commit 52c66cd63dafff93ac689aa9b3952a9b7a451cb3
Author: Ender ÇELİK <celikk.ender@gmail.com>
Date: Sun Dec 17 19:35:47 2023 +0300

    initial commit
```

pull:

- Uzak bir depodan (remote repository) güncellemeleri çekmek ve yerel depoyu güncellemek için kullanılır.
- **git pull** komutu, aslında **git fetch** ve ardından **git merge** işlemlerini birleştirerek gerçekleştirir.



push:

- Yerel (local) depodaki değişiklikleri uzak (remote) depoya yüklemek için kullanılır.
- Yerelde yapılan commit'leri, uzak depoya gönderir ve böylece projenin güncel versiyonunun paylaşılmasına olanak tanır.
- **Push** işlemi yapmadan önce, uzak depoda yapılmış olabilecek güncellemeleri almak için **git pull** komutunu kullanmak iyi bir uygulamadır.

fork:

- “**fork**” bir depoyu (repository) kopyalama işlemidir.
- Orijinal projenin sizin kontrolünüzdeki bağımsız kopyasını oluşturmayı sağlar.
- Fork edilen depo, tamamen sizin kontrolünüz altında bir kopyadır ve orijinal projeden bağımsızdır.

fork:

- İlk olarak fork edilmek istenen depo ilgili platformdaki (GitHub, GitLab vb...) kişisel hesaba kopyalanır.



- Daha sonra kopyalanan depo lokale alınır.

```
C:\Users\Ender\Desktop\vcs_presentation>git clone https://github.com/necmettinalver/Exploratory-Data-Analysis.git
```

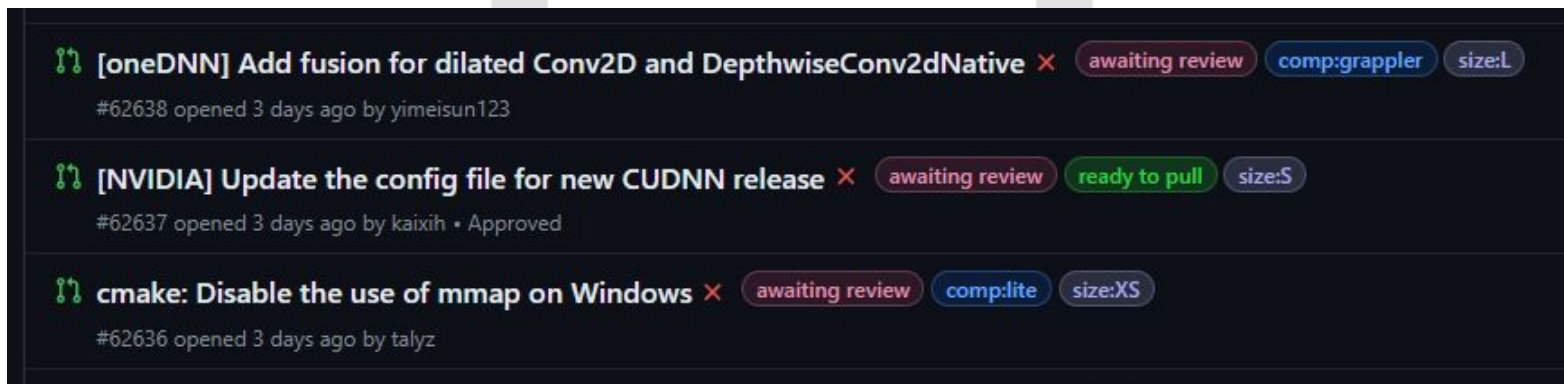
pull request (PR):

- Fork edilmiş bir depodaki değişikliklerin orijinal depoya entegre etme talebini ifade eder.
- Açık kaynak yazılım geliştirme işleyişinde önemli bir yer tutar.



pull request (PR):

- Örneğin: GitHub'da ki TensorFlow repository'sine atılan pull request'ler:
- **“contributor”**



Bölüm 3:

Git ile Örnek Proje Yönetimi

- Bu bölümde kullanıcı bilgileri tanımlama işleminden başlayarak bir repo yönetiminde kullanılabilecek giriş-orta-ileri(kısmen) seviye git komutlarını temsili bir proje üzerinde uygulayarak öğreneceğiz.

Install & config:

- Git versiyon kontrol sistemini linux terminalden sistemimize yükleyelim:

```
:~$ sudo apt-get install git
```

- **config** komutu ile mail adresimizi ve adımızı git'e tanımlayalım:
- **-global** parametresi ile bu tanımlamayı tamamen sisteme, • veya sadece o repo'ya özel olarak tanımlayabiliriz.

```
$ git config --global user.email "celikk.ender@gmail.com"  
$ git config --global user.name "Ender CELIK"
```

```
$ git config user.email "celikk.ender@gmail.com"  
$ git config user.name "Ender CELIK"
```

initialize:

- “**git init**” komutu ile bulunduğumuz konumda bir git reposu oluşturarak başlayalım:

```
C:\Users\Ender\Desktop\vcs_presentation>git init  
Initialized empty Git repository in C:/Users/Ender/Desktop/vcs_presentation/.git/
```

- **.gitignore** dosyamızı oluşturarak. Hangi uzantı, dizin veya dosyaları takip etmeyeceğimizi sisteme bildirelim.

.gitignore:

- “.user” uzantılarını, /node_modules ve /build dizinlerini, “.env” adında API anahtarlarımızı tutacağımız dosyayı, takip sistemine dahil etmediğimizi bildiren .gitignore dosyamızın görünümü bu şekilde:

```
#extension
*.user

#path
/node_modules
/build

#specific files
.env
```

- Son durumda dosyalarımız ise böyle gözüküyor:

******GitHub repo oluştururken dil bazlı olarak bir .gitignore dosya taslağı veriyor.

Ad

- .git
- build
- node_modules
- .env
- .gitignore
- MainPage.js
- NewOrderPage.js
- ProfilePage.js

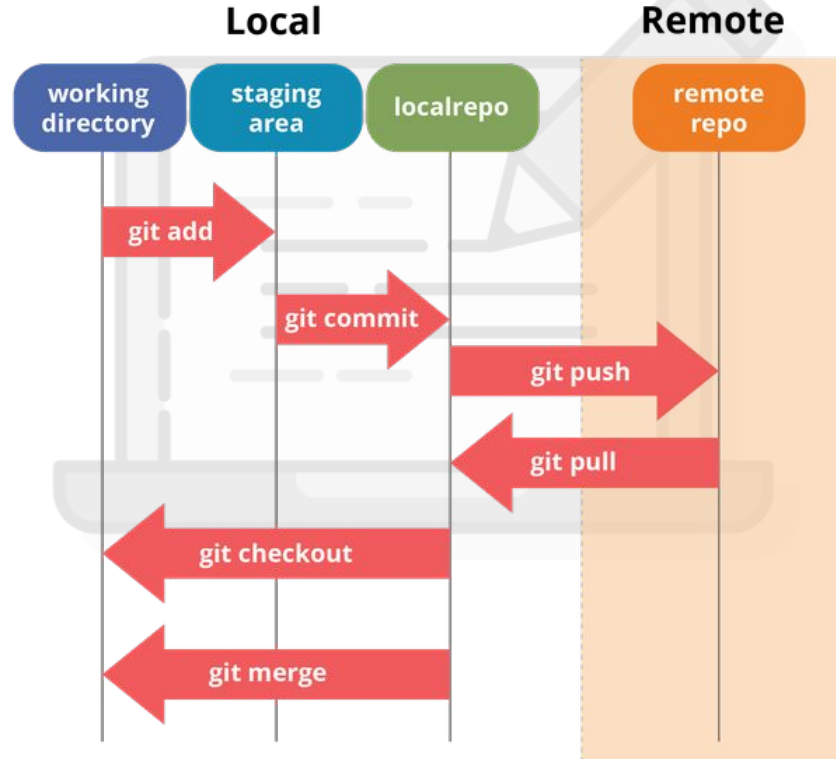
add:

- İlk commit'imizi oluşturmak için “**add**” komutu ile dosyalarımızı **staging area**'ya ekleyelim.

***Staging Area:**

- Commit oluşturmadan önce commit'e dahil edilecek düzenlemeleri kontrol edebileceğimiz bir alan.
- Dosyalarımızı “**git add <degisken_parametre>**” ile önce staging area'ya dahil edip, burada son kontrolleri yaptıktan sonra commit işlemini gerçekleştiririz.

staging area:



*git reset & git status

add:

- “**add**” komutunu farklı şekillerde kullanabiliriz.
- “**git add .**” komutu ile .gitignore dosyasında belirttiğim uzantı, izin ve isimdekiler hariç bütün dosyaları,
- “**git add 'MainPage.js'**” komutu ile sadece MainPage.js adlı dosyayı,
- “**git add /path**” komutu ile o dizini,
- “**git add *.js**” komutu ile bütün .js uzantılı dosyaları staging area'ya dahil edebiliriz.

add:

```
C:\Users\Ender\Desktop\vcs_presentation>git add .
```

- Bütün dosyalarımızı staging area'ya ekledik.

status & reset:

- “**git status**” ile staging area’mızı kontrol edelim ve örneğin “NewOrderPage.js” dosyasını oluşturmayı planladığımız commit’in dışında bırakmak üzere staging area’dan çıkaralım:

```
C:\Users\Ender\Desktop\vcs_presentation>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .gitignore
    new file:   MainPage.js
    new file:   NewOrderPage.js
    new file:   ProfilePage.js
    new file:   tools/ProjectsTools.js
```

```
C:\Users\Ender\Desktop\vcs_presentation>git reset NewOrderPage.js
```

commit:

- -m (message) parametresini vererek ilk commit'imizi atalım.

```
C:\Users\Ender\Desktop\vcs_presentation>git commit -m "initial commit"
[main (root-commit) eda7504] initial commit
5 files changed, 9 insertions(+)
create mode 100644 .gitignore
create mode 100644 MainPage.js
create mode 100644 NewOrderPage.js
create mode 100644 ProfilePage.js
create mode 100644 tools/ProjectsTools.js
```

- -m (message) parametresi hiç vermeyip komutu çalıştırmamızla açılacak sayfayı da tamamen açıklama sayfası olarak kullanabiliyoruz.

log:

- Farklı şekillerde geçmiş commit'lerimizi görüntüleyelim.

git log

```
C:\Users\Ender\Desktop\vcs_presentation>git log
commit eda75044850b2de6f9243a64a1f31bcd1a2f3f28 (HEAD -> main)
Author: Ender ÇELİK <celikk.ender@gmail.com>
Date:   Sun Dec 17 23:37:16 2023 +0300

    initial commit
```

log:

git log -stat

```
C:\Users\Ender\Desktop\vcs_presentation>git log --stat
commit eda75044850b2de6f9243a64a1f31bcd1a2f3f28 (HEAD -> main)
Author: Ender ÇELİK <celikk.ender@gmail.com>
Date:   Sun Dec 17 23:37:16 2023 +0300

    initial commit

 .gitignore      | 9 ++++++++
 MainPage.js     | 0
 NewOrderPage.js | 0
 ProfilePage.js  | 0
 tools/ProjectsTools.js | 0
 5 files changed, 9 insertions(+)
```

git log -oneline

```
C:\Users\Ender\Desktop\vcs_presentation>git log --oneline
eda7504 (HEAD -> main) initial commit
```

commit:

- Bazı durumlarda commit'lerimiz arasında geçiş yapmamız gerekebilir.
- **"git checkout HEAD^"** veya **"git checkout HEAD~"** ile commit tanımını ve düzenlemelerimizi silerek bir önceki commit'e geçebiliriz.
- **"git checkout HEAD^ <sayi_degeri>"** ile parametre adedince sayarak, commit tanımlarını ve düzenlemelerimizi silerek x önceki commit'e geçebiliriz.
- **"git checkout HEAD^ <commit ID>"** ile id'sini verdiğimiz commit'ten sonraki commit'lerin commit tanımlarını ve düzenlemelerimizi silerek o commit'e geçebiliriz.
- Yukarıdaki değişimlerden sonra **"git switch -"** komutu ile bu işlemleri geri alabiliriz.

commit:

- “**git reset –soft <commit ID>**” komutu ile düzenlemelerimiz bizde kalacağı ve staging area’ya ekleneceği şekilde id’sini verdiğimiz commit’e geçebiliriz.
- “**git reset –hard <commit ID>**” komutu ile sonraki commit’ler tamamen silinecek şekilde id’sini verdiğimiz commit’e geçebiliriz.

****** Bu değişimlerden sonra geri alma işlemi yapılamaz!

branch:

- Yeni bir özellik eklemek istediğimizi düşünelim ve bunun için bir “feature branch” oluşturmak istediğimizi varsayalım.
- Üzerinde olduğumuz branch’i ve varsa diğer branch’lerimizi görelim:

```
C:\Users\Ender\Desktop\vcs_presentation>git branch
* main
```


branch:

- “feature branch” adlı branch’imizi oluşturalım.

```
C:\Users\Ender\Desktop\vcs_presentation>git branch feature-branch  
  
C:\Users\Ender\Desktop\vcs_presentation>git branch  
feature-branch  
* main
```

- Yeni oluşturduğumuz branch’e geçiş yapalım. (switch veya checkout)

```
C:\Users\Ender\Desktop\vcs_presentation>git checkout feature-branch  
Switched to branch 'feature-branch'
```

branch:

- Ayrıca bu şekilde oluşturma ve branch'e geçme işlemini aynı anda gerçekleştirebilirdik.

```
C:\Users\Ender\Desktop\vcs_presentation>git checkout -b feature-branch  
Switched to a new branch 'feature-branch'
```

- Projemize özellik eklemek için oluşturduğumuz “feature-branch” adlı branch'te geliştirmeler yaptığımız ve bu geliştirmeleri o branch içerisinde commit'lediğimiz aşamaları geçerek main (ana) branch'imiz ile üzerinde çalıştığımız branch'i birleştirme, **merge** etme kısmına geçiyoruz.

merge:

- En basit tanımıyla farklı dallardaki değişiklikleri birleştirme işlevi sağlayan **merge** komutu ile dallarımızı birleştirelim:

```
C:\Users\Ender\Desktop\vcs_presentation>git merge feature-branch
Updating ee0a3d3..ac5bb70
Fast-forward
 MainPage.js | 2 +-
 ProfilePage.js | 4 ++++
 2 files changed, 5 insertions(+), 1 deletion(-)
```

- Birleştirme işlemi her zaman bunun gibi kusursuz olmayıp, çakışmalar içerebilir.

***Conflict Resolution: (Çakışmaların Çözülmesi)**

merge:

- Aynı satırı iki branch'imizde farklı şekilde düzenleyelim ve çakışma (conflict) durumunu gözlemleyelim.

```
1 git presentation. -- main branch edit conflict line --
```

```
1 git presentation. -- feature-bracnh edit --
```

******Genellikle bu çakışmalar main'den referans alan diğer branch'lerin main üzerindeki birleşmesi esnasında gerçekleşir.

Merge işlemi nasıl gerçekleşecek?

- Git çakışmayı belirleyecek ve bizden müdahale bekleyecek.

merge:

- **git merge feature-branch** , komutunun çıktısı:

```
C:\Users\Ender\Desktop\vcs_presentation>git merge feature-branch
Auto-merging MainPage.js
CONFLICT (content): Merge conflict in MainPage.js
Automatic merge failed; fix conflicts and then commit the result.
```

- MainPage.js dosyasının son durumunu görüntüleyelim:

```
C:\Users\Ender\Desktop\vcs_presentation>type MainPage.js
<<<<<<< HEAD
git presentation. -- main branch edit conflict line --
=====
git presentation. -- feature-bracnh edit --
>>>>>>> feature-branch
```

merge:

- Bu noktada üç seçeneğimiz var. Dış branch'ten gelen düzenlemeyi alabilir, mevcut main'deki düzenlemeyi koruyabilir veya her ikisini de alabiliriz.

```
C:\Users\Ender\Desktop\vcs_presentation>git status
On branch main
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   MainPage.js

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\Ender\Desktop\vcs_presentation>git add MainPage.js

C:\Users\Ender\Desktop\vcs_presentation>git commit -m "merge"
[main 1ffdf7f] merge

C:\Users\Ender\Desktop\vcs_presentation>git merge feature-branch
Already up to date.

C:\Users\Ender\Desktop\vcs_presentation>type MainPage.js
git presentation. -- main branch edit conflict line --
git presentation. -- feature-bracnh edit --
```

merge:

- Ek branch ile işlemiz bittiğine göre onu silebiliriz:

```
C:\Users\Ender\Desktop\vcs_presentation>git branch
feature-branch
* main

C:\Users\Ender\Desktop\vcs_presentation>git branch -d feature-branch
Deleted branch feature-branch (was d0d0a9b).

C:\Users\Ender\Desktop\vcs_presentation>git branch
* main
```

- Silmek istediğimiz branch'te commit edilmemiş değişiklikler var ise silme işlemi `-D` parametresi ile gerçekleştirilir.

merge:

- Normal yönetimde oluşan çakışma(conflict) durumu ve görünümü:

```
C:\Users\Ender\Desktop\vcs_presentation>git branch
  feature/branch
  fix/branch
* main

C:\Users\Ender\Desktop\vcs_presentation>git merge fix/branch
Updating dcec96d..fc01a2b
Fast-forward
 ProfilePage.js | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\Ender\Desktop\vcs_presentation>git merge feature/branch
Auto-merging ProfilePage.js
CONFLICT (content): Merge conflict in ProfilePage.js
Automatic merge failed; fix conflicts and then commit the result.
```


merge:

- Genellikle oluşan çakışma durumu ve görünümü: (* rebase komutu)

```
> Users > Ender > Desktop > vcs_presentation > JS ProfilePage.js
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<<<< HEAD (Current Change)
2 fix
3 =====
4 feature
5 >>>>>>> feature/branch (Incoming Change)
6 ProfilPage.js => initial commit line 2
7 ProfilPage.js => initial commit line 2
```

remote:

- Temsili repomuzu oluşturduk, commit'ler attık, branch'ler ile geliştirmeler/hata gidermeleri yaptık ve bu düzenlemelerimizi merge komutu ile main branch'e entegre etme işlemlerini gerçekleştirdik.
- Şimdi de Git'in uzak repository ilişkilerini inceleyelim.



Bitbucket



GitHub:

- Projeye bulutta depolama ve erişim imkanı sağlar.
- Private ve Public repo seçenekleri ile projenin erişilebilirlik kapsamını belirleme imkanı vardır. (2019)
- Public repository'leri ile açık kaynak yazılım geliştirmeye ev sahipliği yapan öncü platformlardan biri konumunda.



GitHub

remote:

- Github hesabımızda bir repository oluşturalım ve bu repository'e lokalde daha önce üzerinde çalıştığımız repo'yu yüklemek isteyelim.
- *birden fazla uzak repo bağlantısı yapabiliriz: origin_github, origin_bitbucket

```
git remote add origin https://github.com/enderceliik/vcs-presentetion.git
git branch -M main
git push -u origin main
```

remote:

- Sırasıyla uygulayacağımız adımlar sonrasında terminalimiz böyle gözükecek.

```
C:\Users\Ender\Desktop\vcs_presentation>git remote add origin https://github.com/enderceliik/vcs-presentation.git
C:\Users\Ender\Desktop\vcs_presentation>git branch -M main
C:\Users\Ender\Desktop\vcs_presentation>git push -u origin main
Enumerating objects: 40, done.
Counting objects: 100% (40/40), done.
Delta compression using up to 4 threads
Compressing objects: 100% (35/35), done.
Writing objects: 100% (40/40), 3.33 KiB | 227.00 KiB/s, done.
Total 40 (delta 21), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (21/21), done.
To https://github.com/enderceliik/vcs-presentation.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

remote & pull:

- Projemizde geliştirmeler yapalım ve bunları uzak depomuza dahil edelim.
- Bu proje üzerinde çalışan başka geliştiriciler de var olduğunu kabul ediyoruz ve biz geliştirmelerimizi entegre etmeden önce güncel halini çekmemiz gerek.

```
C:\Users\Ender\Desktop\vcs_presentation>git pull  
Already up to date.
```

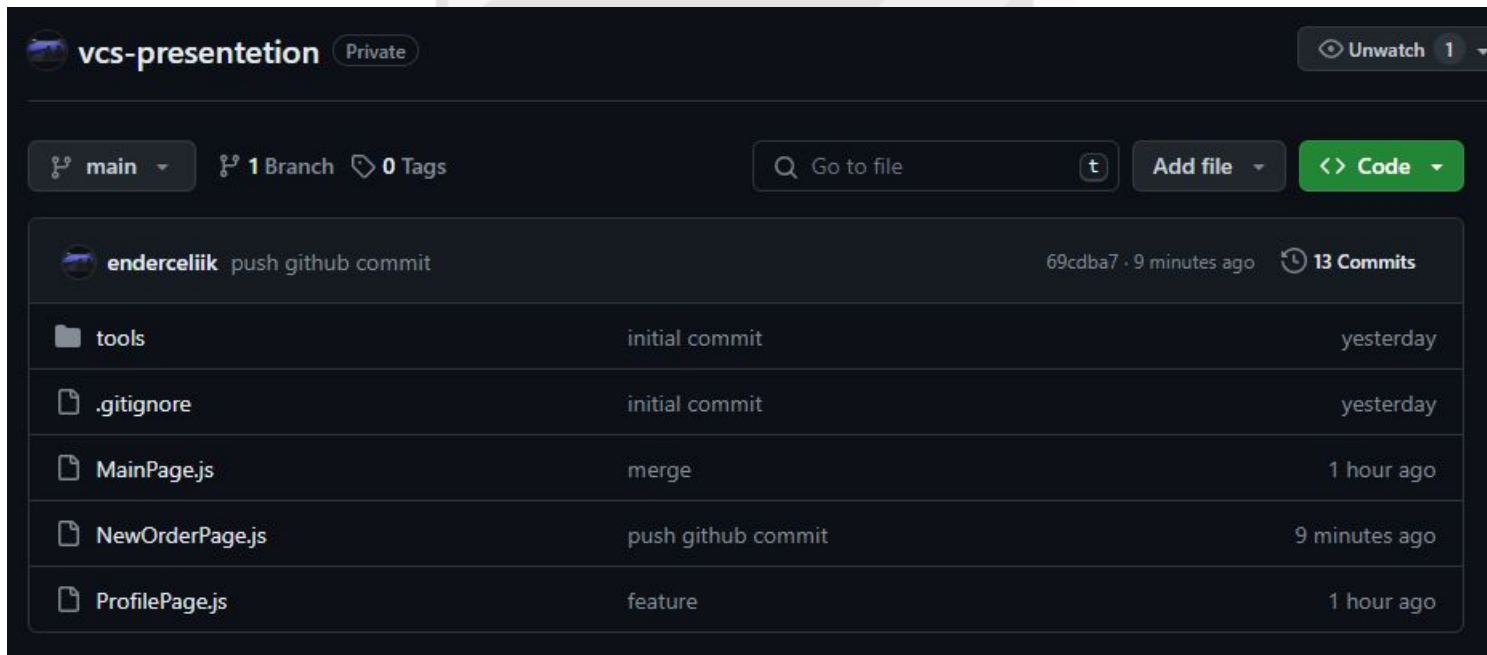
remote & push:

- Şimdi **push** edebiliriz.

```
C:\Users\Ender\Desktop\vcs_presentation>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 381 bytes | 190.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/enderceliik/vcs-presentetion.git
   c21c798..69cdba7  main -> main
```

remote:

- GitHub görünüm:



The screenshot shows the GitHub interface for a repository named 'vcs-presentation'. The repository is marked as 'Private'. At the top, there are buttons for 'main' (selected), '1 Branch', and '0 Tags'. A search bar 'Go to file' is present, along with 'Add file' and 'Code' buttons. Below this, a commit by 'enderceliik' is shown, dated '9 minutes ago' with commit hash '69cdba7'. The commit message is 'push github commit'. Below the commit, a list of files is displayed:

File	Commit Message	Time
tools	initial commit	yesterday
.gitignore	initial commit	yesterday
MainPage.js	merge	1 hour ago
NewOrderPage.js	push github commit	9 minutes ago
ProfilePage.js	feature	1 hour ago

feature branch:

- Şimdi de projemize yeni bir özellik eklemek istediğimizi varsayalım.
- Sırasıyla yapmamız gerekenler:
 - Adlandırma standardına uymak kaydıyla bir özellik branch'i oluşturmak.
 - Bu branch'e geçiş yapıp yeni özellik çalışmasını gerçekleştirmek.
 - Çalışma sonucunda commit'ler atıp bu branch'i uzak depoya push'lamak.

feature branch:

- Branch'imizi oluşturup geçiş yapalım:

```
C:\Users\Ender\Desktop\vcs_presentation>git checkout -b features/add_readmeMD_file  
Switched to a new branch 'features/add_readmeMD_file'
```

- Özelliğimizi ekledik. Commit'imizi atalım.

```
C:\Users\Ender\Desktop\vcs_presentation>git commit -m "readme.md dosyası eklendi."  
[features/add_readmeMD_file 4033204] readme.md dosyası eklendi.  
1 file changed, 7 insertions(+)  
create mode 100644 README.md
```

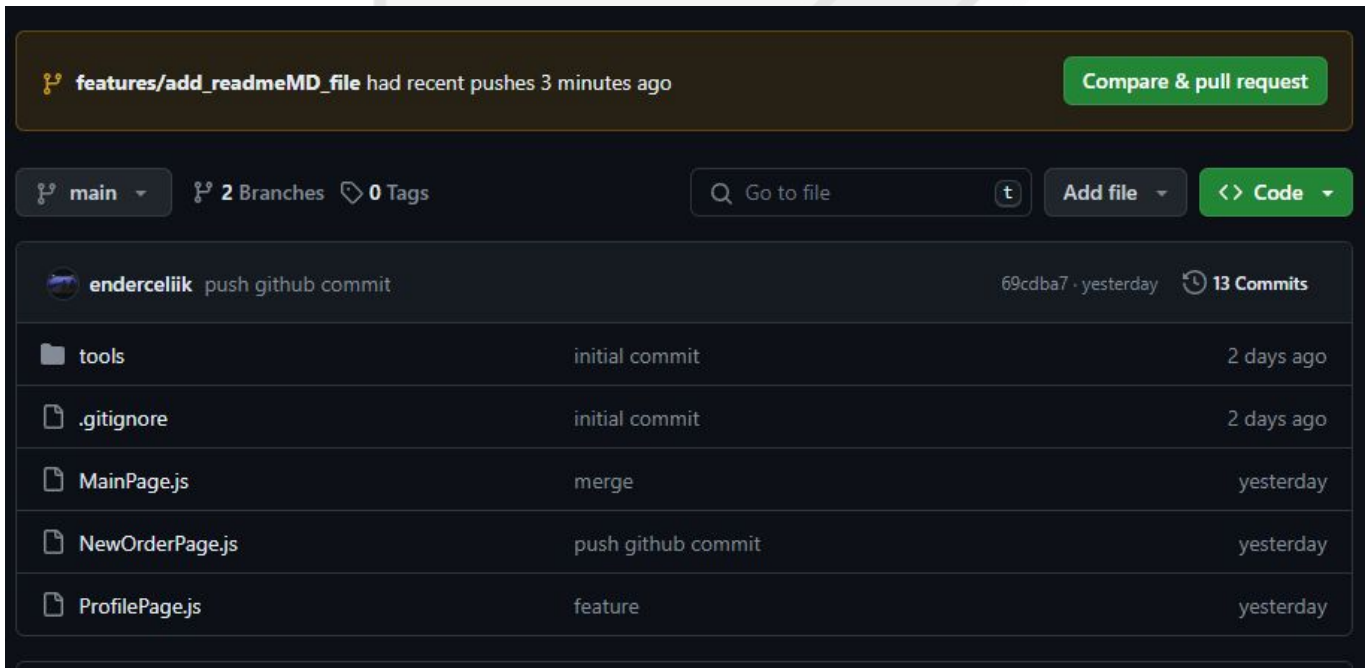
feature branch:

- Bu özellik için özel olarak oluşturduğumuz branch'i uzak depoya push'layalım:

```
C:\Users\Ender\Desktop\vcs_presentation>git push -u origin features/add_readmeMD_file
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 481 bytes | 240.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'features/add_readmeMD_file' on GitHub by visiting:
remote:   https://github.com/enderceliik/vcs-presentation/pull/new/features/add_readmeMD_file
remote:
To https://github.com/enderceliik/vcs-presentation.git
 * [new branch]      features/add_readmeMD_file -> features/add_readmeMD_file
branch 'features/add_readmeMD_file' set up to track 'origin/features/add_readmeMD_file'.
```

feature branch:

- GitHub görünüm:




The screenshot shows a GitHub repository interface. At the top, a notification bar indicates that the branch 'features/add_readmeMD_file' has recent pushes 3 minutes ago, with a 'Compare & pull request' button. Below this, the repository navigation bar shows 'main' as the selected branch, with '2 Branches' and '0 Tags' available. A search bar labeled 'Go to file' and buttons for 'Add file' and '<> Code' are also present. The main content area displays a list of files and their commit history:

File	Commit Message	Commit Hash	Time
tools	initial commit	69cdba7	2 days ago
.gitignore	initial commit		2 days ago
MainPage.js	merge		yesterday
NewOrderPage.js	push github commit		yesterday
ProfilePage.js	feature		yesterday

The repository is owned by 'enderceliik' and has 13 commits in total.

feature branch:

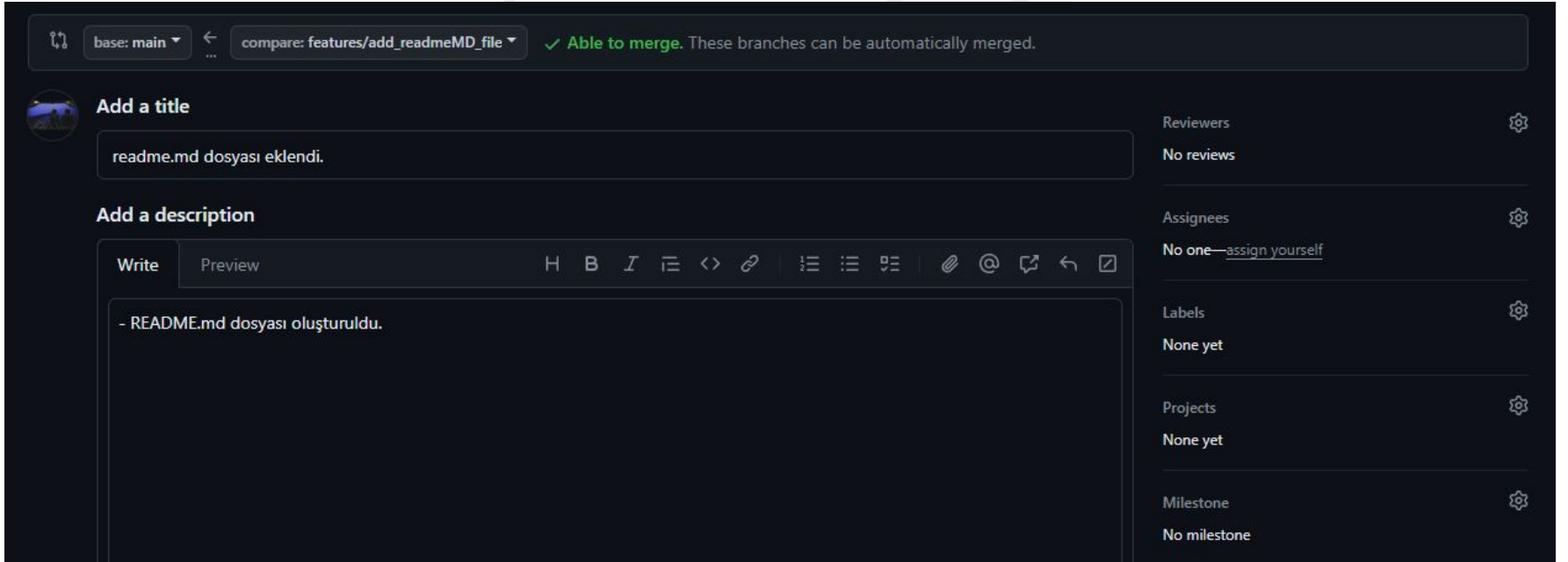
- Uzak depomuzda pull request yaparak yeni özelliği master/main branch'ine entegre edelim:

 **features/add_readmeMD_file** had recent pushes 18 minutes ago

[Compare & pull request](#)

pull request:

- Pull request ekranı



The screenshot shows a GitHub Pull Request interface in dark mode. At the top, there's a header bar with a fork icon, 'base: main', a dropdown arrow, 'compare: features/add_readmeMD_file', and a green checkmark indicating 'Able to merge. These branches can be automatically merged.' Below this, on the left, is a profile picture of a person with a blue background. To the right of the profile picture is the 'Add a title' section with a text input field containing 'readme.md dosyası eklendi.' Below the title section is the 'Add a description' section with a 'Write' tab selected and a 'Preview' tab. The 'Write' tab has a rich text editor toolbar with icons for bold, italic, link, list, code, quote, and other formatting options. The text area contains '- README.md dosyası oluşturuldu.' On the right side of the interface, there are several sections: 'Reviewers' with 'No reviews', 'Assignees' with 'No one' and a link to 'assign yourself', 'Labels' with 'None yet', 'Projects' with 'None yet', and 'Milestone' with 'No milestone'. Each of these sections has a gear icon for settings.

base: main compare: features/add_readmeMD_file ✓ Able to merge. These branches can be automatically merged.

Add a title

readme.md dosyası eklendi.

Add a description

Write Preview

H B I

- README.md dosyası oluşturuldu.

Reviewers

No reviews

Assignees

No one [assign yourself](#)

Labels

None yet

Projects

None yet

Milestone

No milestone

pull request:

- Reviewers: Kod incelemelerini yapacak ve değişiklikleri onaylayacak veya reddedecek olan geliştiriciler.
- Assignees: Bu pull request'ten sorumlu geliştiriciler.

Reviewers



No reviews

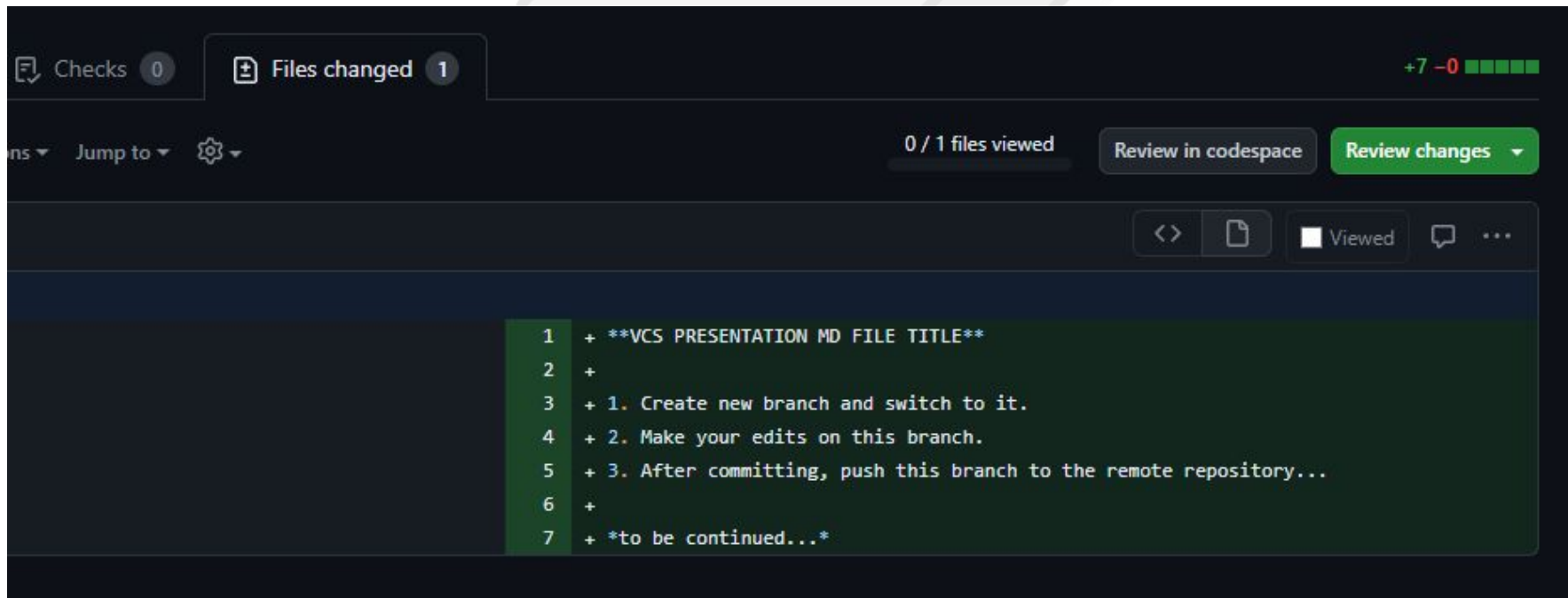
Assignees



enderceliik

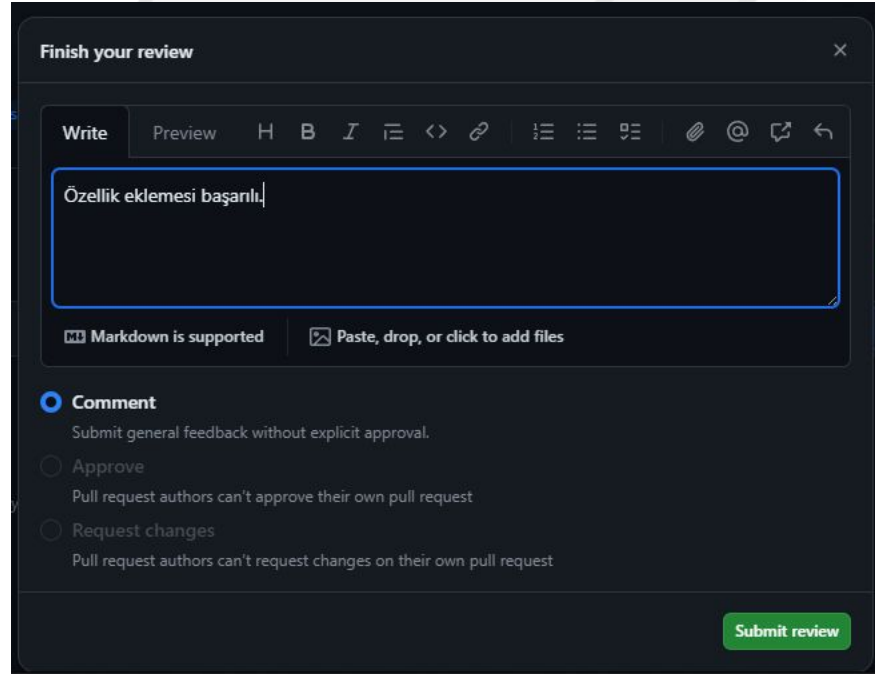
pull request:

- İnceleme ve onay verme:



pull request:

- İnceleme ve onay verme:



Finish your review

Write Preview H B I ≡ < > 🔗 ≡ ≡ ≡ 📎 @ ↻ ↶

Özellik eklemesi başarılı

📄 Markdown is supported 📎 Paste, drop, or click to add files

☒ **Comment**
Submit general feedback without explicit approval.

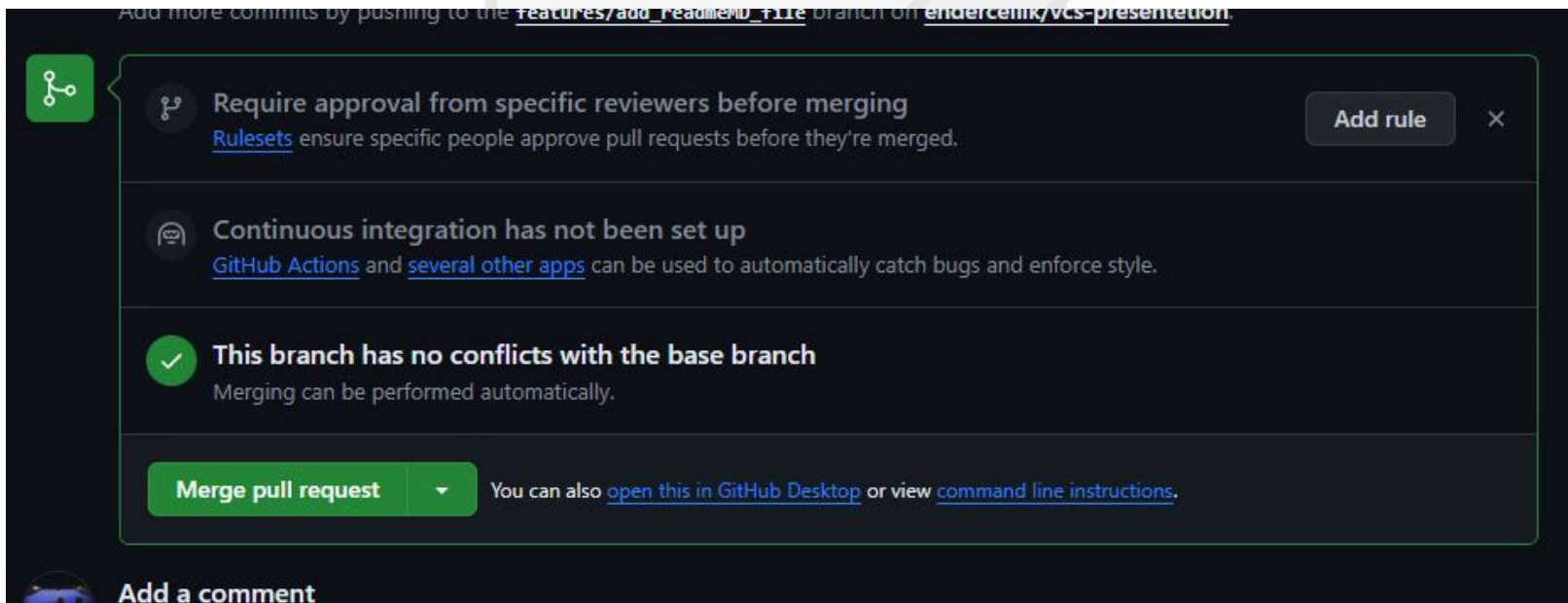
☐ Approve
Pull request authors can't approve their own pull request

☐ Request changes
Pull request authors can't request changes on their own pull request

Submit review

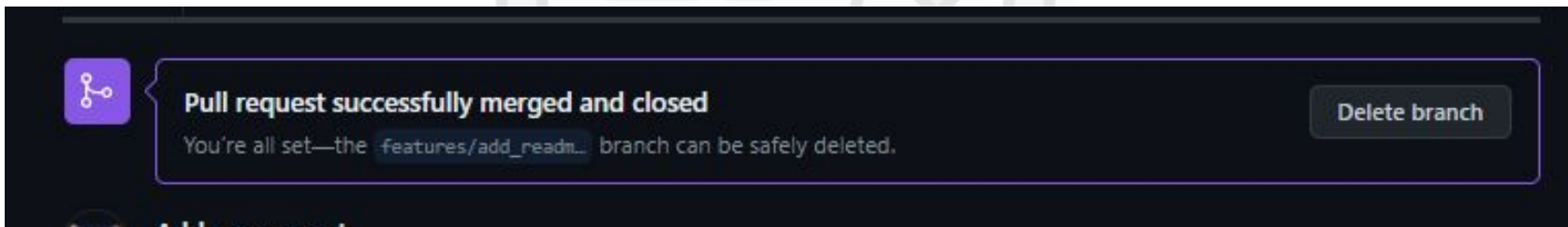
pull request:

- Pull request'i kabul etme ve master ile merge etme:



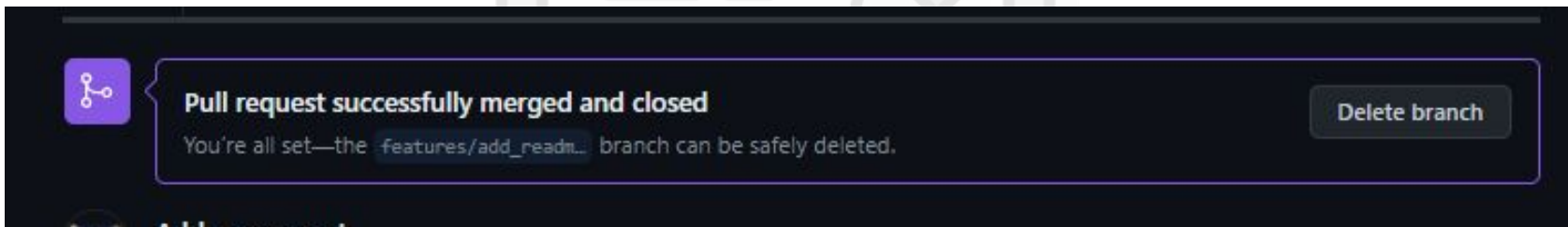
pull request:

- Pull request'i kabul etme ve master ile merge etme:



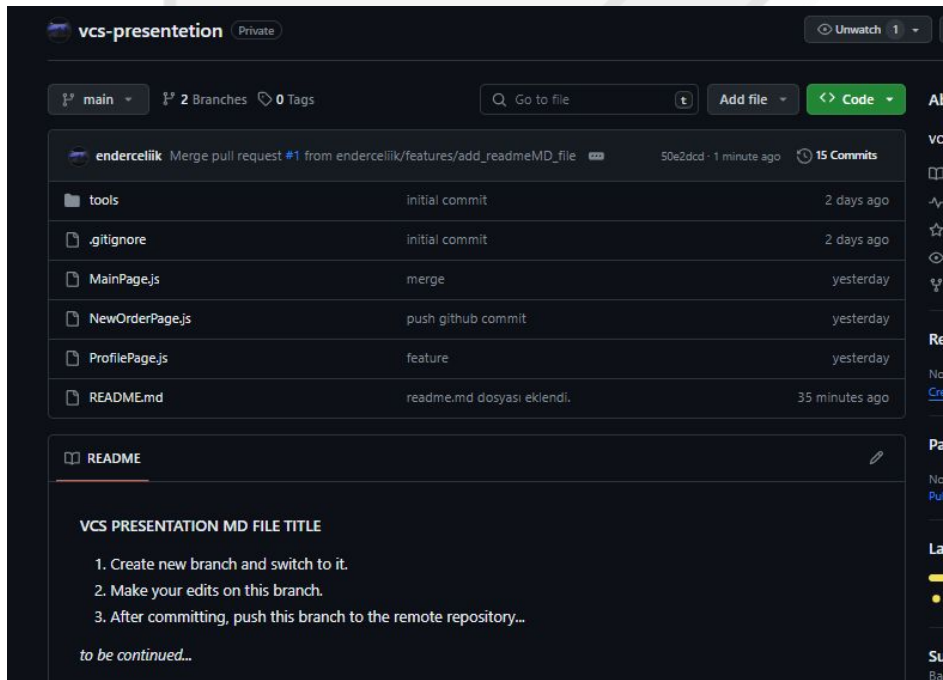
pull request:

- Pull request'i kabul etme ve master ile merge etme:



pull request:

- GitHub görünüm:



feature branch:

- main'e geçip uzak repository'i pull ediyoruz ve bir branch döngüsü tamamlanmış oluyor.

```
C:\Users\Ender\Desktop\vcs_presentation>git branch
* features/add_readmeMD_file
  main

C:\Users\Ender\Desktop\vcs_presentation>git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\Ender\Desktop\vcs_presentation>git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 666 bytes | 83.00 KiB/s, done.
From https://github.com/enderceliik/vcs-presentation
   69cdba7..50e2dcd  main       -> origin/main
Updating 69cdba7..50e2dcd
Fast-forward
 README.md | 7 ++++++
 1 file changed, 7 insertions(+)
 create mode 100644 README.md
```

***Vakit ayırdığınız
için Teşekkürler!***