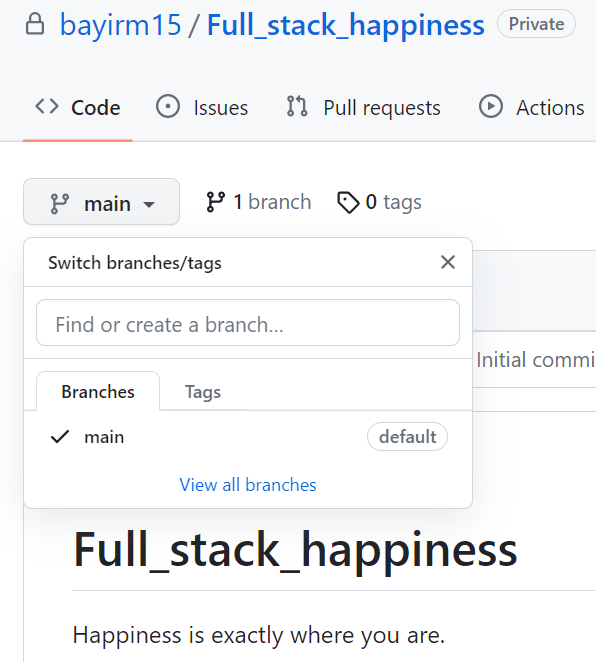
GİT Uygulamalı Örneği:

## Uzak sunucuda oluşturduğumuz repomuzu yerel dizinimize kopyalayalım(clone’lamak).



Bir repo oluşturduğunuzda “Main” veya “Master” dal varsayılan daldır. Yukarıda görüldüğü üzere, GitHub varsayılan dal olarak main' i kullanıyor. “Main” bizim birleştirilme(merging) dalımız olacak.

Daha sonra göreceğiz ancak “merging” için kısa bir tanım yapmamız gerekir ise:

Birleştirme(Merging), değişiklikleri bir branchten alma ve bunları diğerine (genelde “Main” branch) uygulamadır.

Aşağıdaki iki satır komutu GİT BASH üzerinde çalıştırdığımızda klonlama işlemi gerçeklemiş olacak:

$ cd "C:\Users\Asus\Desktop\Musa\_All\Genel Eğitim Dokümanları\Deneme"

$ git clone -b main https://github.com/bayirm15/Full\_stack\_happiness.git

Repomuz ile birlikte gelen tüm branchleri listeleyelim:

$ git branch -a

\* main

remotes/origin/main

“main” branch başında \* işareti var. Bu şu an aktif olarak o branchtesin anlamına geliyor.

## Yeni bir dal(branch) yaratalım ve değişikliklerimizi o dal üzerinde yapalım.

Değişikliklerimizi “main” branch harici geçici bir dal yaratarak onun üzerinde yapmalıyız. Sonrasında bu değişiklikleri ekip arkadaşlarımızdan kontrol ve onay alarak “main” branch ile birleştireceğiz.

Aşağıdaki checkout komutu ile yeni bir branch yaratıyoruz(.git dizinimizin olduğu yerde) ve branch’ i aktif dalımız haline getiriyoruz:

$ cd "C:\Users\Asus\Desktop\Musa\_All\Genel Eğitim Dokümanları\Deneme\Full\_stack\_happiness"

$ git checkout -b test

Aktif tüm branchleri listeleyelim:

$ git branch –a

main

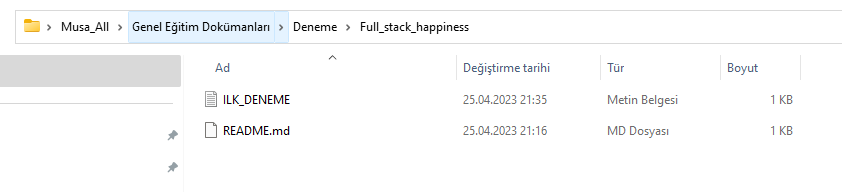
\* test

remotes/origin/main

“test” branch başında \* işareti var. Bu şu an aktif olarak o branchtesin anlamına geliyor.

## Çalışma dizinimize bir metin dosyası ekleyelim ve nasıl bir süreçten geçeceğini birlikte takip edelim (tüm dosya ekleme-çıkarma-güncelleme işlemlerinizi yapabilirsiniz.)

“test” dalında olduğumuza göre artık çalışma dizinimizde istediğimiz değişiklikleri yapabiliriz.



“ILK\_DENEME” adında bir metin dosyasını çalışma dizinimize ekledik.

Üzerinde çalıştığımız “test” dalında yapılan değişiklikleri görmek için aşağıdaki komutu çalıştırıyoruz:

$ git status

On branch test

Untracked files:

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

ILK\_DENEME.txt

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

Görüldüğü üzere GİT değişiklikleri fark etti. Ancak bu noktada bunun ile pek ilgilenmeyecek. Bize sadece “tamamdır, çalışan ağaçta yeni bir dosyanın varlığını gördüm, ancak onu izlemem söylenmedi” diyecektir.

Bir dosya oluşturulmuş olmasına rağmen henüz Commit edilecek bir şey yok. Bunun nedeni dosyanın halen izlenmiyor(untracked) oluşudur.

Dosya değişikliklerimizi .git uzantılı çalışma ağacında yaptık. Fakat bu değişiklikleri commit edebilmemiz için ilk olarak Evreleme Alanına(Staging Area) taşımamız gerekiyor.

Git çalışan ağacın içinde olup biten her şeyin farkındadır, ancak yalnızca izlenmesi söylenen dosyaları izler.

Evreleme alanı, çalışan ağacınız ile Git deponuz arasındaki bir hazırlama alanıdır. Git’ in neyi izlemesi gerektiğini, bir commit oluşturduğunuzda neleri içermesi gerektiğini kontrol etmemizi sağlar. Bu alan genellikle .git dizininde yer alır ve bir sonraki commitinize neyin gireceği ile ilgili bilgileri depolar.

Kısa bir terminoloji:

Commit Nedir?

Bir taahhüt bir dosyaya yapılan bireysel bir değişikliktir. Çalışmanızı kaydetme taahhüdünde bulunduğunuzda Git, yapılan değişikliklerin ve bunları kimin ve ne zaman yaptığının kaydını tutmanıza olanak tanıyan benzersiz bir kimlik oluşturur. Taahhütler genellikle, hangi değişikliklerin yapıldığına dair kısa bir açıklama olan bir taahhüt mesajı içerir.

## Evreleme alanına yaptığımız değişiklikleri gönderelim

git add komutu çalışma ağacındaki değişiklikleri evreleme alanına ekler. Git’ e bir sonraki işlemde belirli bir dosyaya yönelik güncellemeleri dahil etmek istediğinizi söyler. Ancak, git commit’ i çalıştırılına dek değişiklikler gerçekten kaydedilmez.

Tüm değiştirilmiş dosyaları eklemek için aşağıdaki komutu kullanabiliriz:

$ git add .

Belirli bir dosyayı eklemek için de şu komutu kullanın(biz belirli bir dosya için çalıştıralım):

cd "C:\Users\Asus\Desktop\Musa\_All\Genel Eğitim Dokümanları\Deneme\Full\_stack\_happiness"

$ git add ILK\_DENEME.txt

Üzerinde çalıştığımız “test” dalında yapılan değişiklikleri görmek için aşağıdaki komutu çalıştırıyoruz:

$ git status

On branch test

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: ILK\_DENEME.txt

“git add ILK\_DENEME.txt “ komutu sonrasında, Git sadece değişiklikleri görmek ile kalmıyor, aynı zamanda takip ediyor. Bu komut ile Git’ e dosyaları ve içerikleri evreleme alanına eklemesini söyleriz. “ILK \_DENEME” dosyası artık evreleme alanına kaydedilmiştir.

Staging alanı bir kaydetme noktasıdır; henüz bir GİT deposunun parçası değildir. Ancak, Git içeriğin anlık bir görüntüsünü kaydetti ve bize commit edilmeye hazır değişikliklerin olduğunu söylüyor.

Staging alanına dosya eklemek iki amaca hizmet eder:

1. Eklenen dosyaların anlık bir görüntüsünü oluşturma(bir tür kaydetme)
2. Bir commit oluşturur iken dikkate alınacak değişiklikleri işaretleme

## Dosyaları kalıcı olarak işleme işlemi: Commit

Commit eklenmesi, çalışır iken ilerlememizi ve değişikliklerimizi takip etmemize yardımcı olur. Git, her commit değişiklik noktasını veya “kaydetme noktasını” dikkate alır. Bir hata bulur iseniz veya değişiklik ihtiyacınızda projede geri dönebileceğiniz noktadır.

Aşağıdaki komut ile değişikliklerini git reposunda kalıcı hale getirmek istediğimiz dosyayı commit ediyoruz:

$ git commit ILK\_DENEME.txt -m "Yeni bir dosya eklendi"

Üzerinde çalıştığımız “test” dalında yapılan değişiklikleri görmek için aşağıdaki komutu çalıştırıyoruz:

$ git status

On branch test

nothing to commit, working tree clean

Son commit’ imiz sonrası proje geçmişini listeleyelim:

$ git log

Author: bayirm15 <bayirmusa45@gmail.com>

Date: Thu Apr 27 00:34:41 2023 +0300

New commit

commit b8ef5853b5b1dd5c1f56b7321b0ac97197ea511b

Author: bayirm15 <bayirmusa45@gmail.com>

Date: Tue Apr 25 23:11:10 2023 +0300

yeni degisiklik

commit 20a0c24b17fea3c151954a482540ab89c49ca3c2

Author: bayirm15 <bayirmusa45@gmail.com>

Date: Tue Apr 25 01:09:19 2023 +0300

New commit

commit cca29493bede47d751b252c4e71ccb70d949bf5c

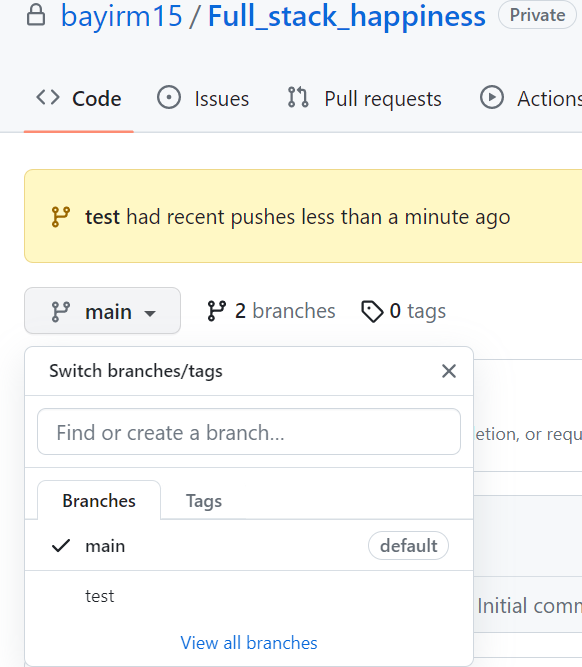
Author: bayirm15 <bayirmusa45@gmail.com>

Date: Wed Apr 19 00:41:30 2023 +0300

## Commit edilen değişikliklerin uzak sunucuya aktarılması:

Bu aşamada commit edilen değişiklikler artık uzak depoya aktarılır. Push etmek deyimi buradan geliyor. git push komutu yerel Git depomuzdan uzak sunucuya commitler göndermemizi sağlar.

$ git push origin test

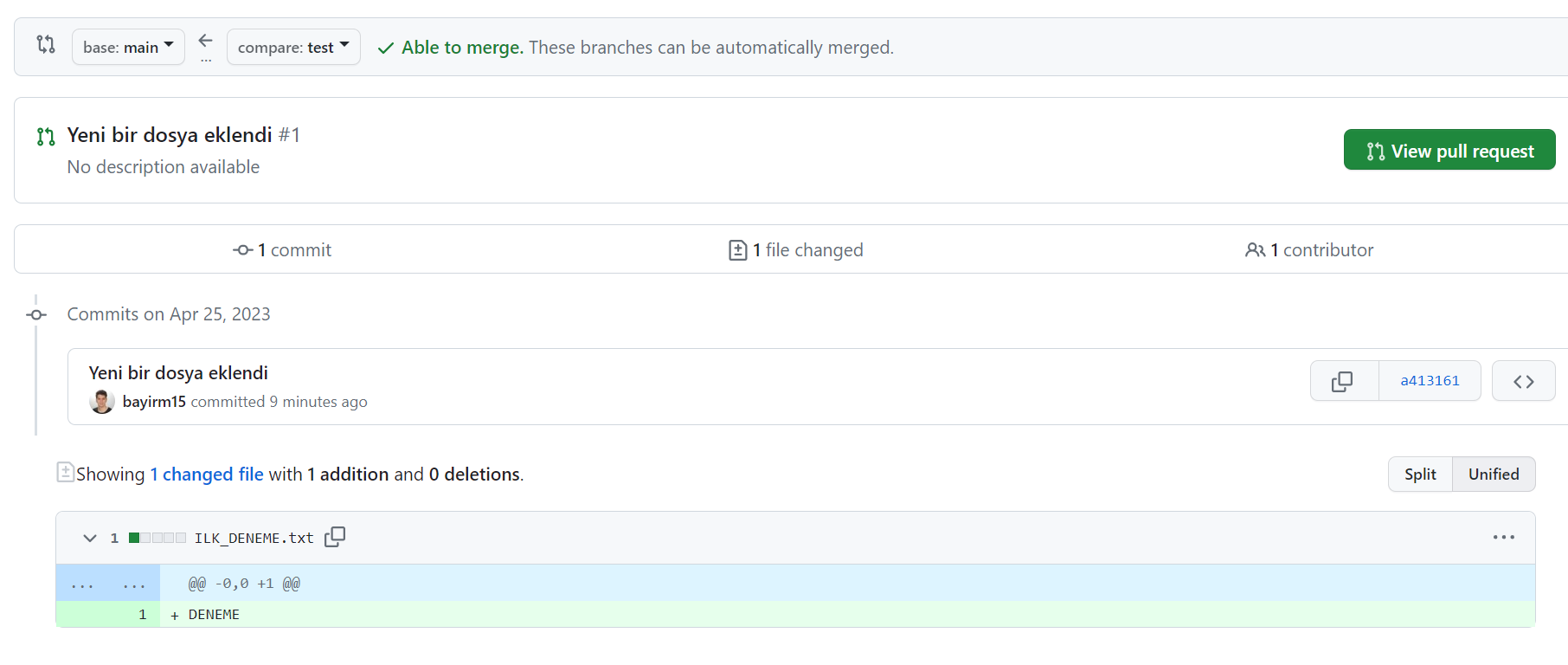


“git push” komutu sonrası Github uzak depomuzda artık iki branch’imizin olduğunu da görüyoruz.

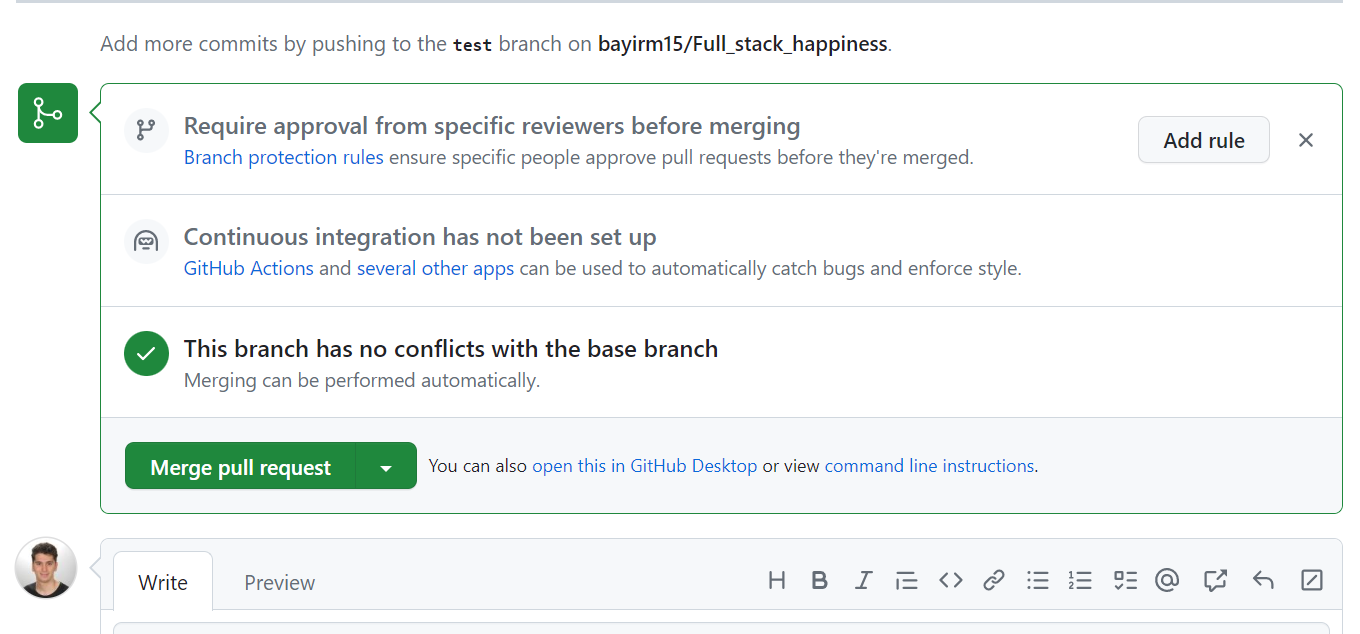
## Main branch ile birleştirme öncesi değişikliğin validasyonu için “Pull Request” talebi oluşturulması:

Çekme isteği, değişikliklerimizin orijinal(main) havuz ile olan karşılaştırmasını gösterir ve hazır olduğunda kodumuzu ana branch ile birleştirir(merge eder).

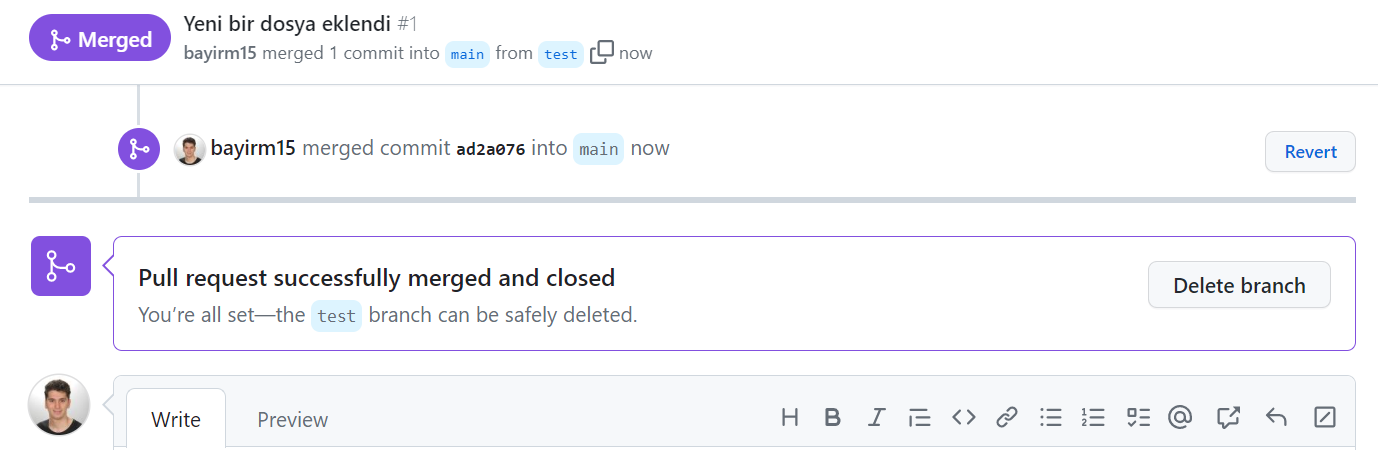
Pull request Github ortamında yaratılabilir:



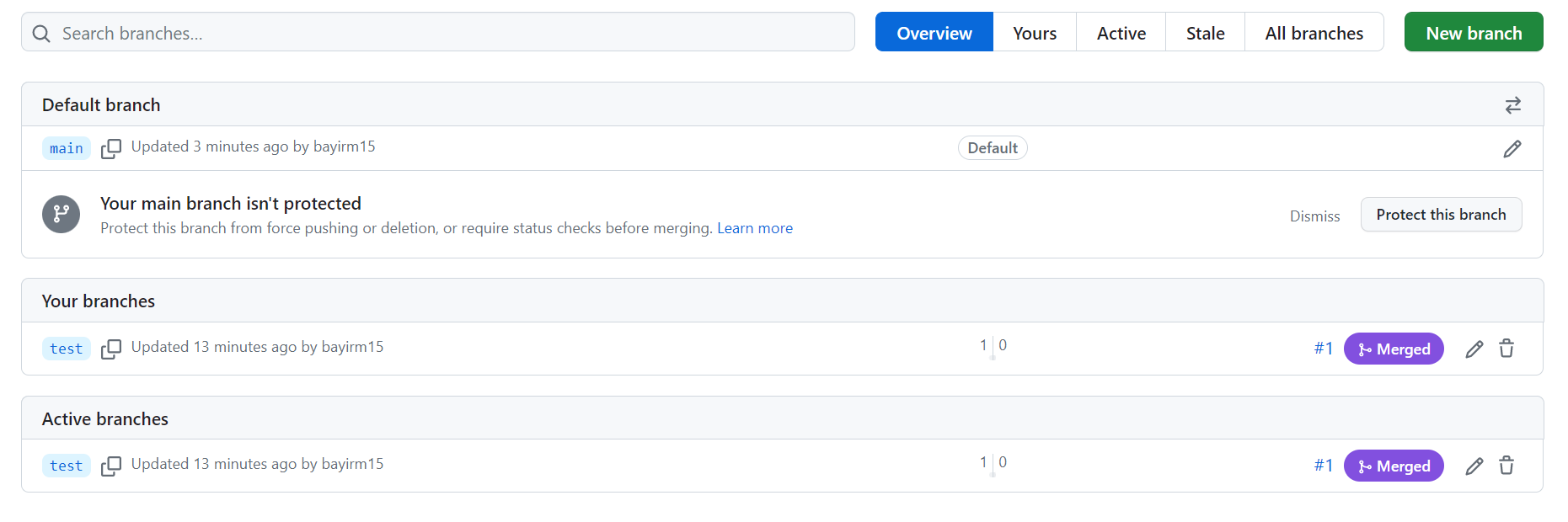
## Çekme isteğine karşı cevaplar olumlu ise artık değişikliklerimiz birleştirme(merging) hazırdır:



Çekme talebi başarılı bir şekilde kapatıldı ve test branch’ inde yaptığımız tüm değişiklikler “main” branch’ e eklenmiş oldu.



## “test” dalı ile artık işimiz kalmadı, depomuzun temiz tutulması için geçici olarak oluşturulan bu branch’ i silebiliriz:



## **Ek adım:**

## Proje üzerinde çalışan geliştiriciler kendi çalışmalarını MAIN dal ile birleştirdikten sonra; değişikliklerin yapıldığı bu uzak depodaki değişmeleri yerel depolarına almalılar.

“git pull”, bir deponun yerel sürümünü uzaktaki bir depodan güncellemek için kullanılan bir Git komutudur.

$ cd "C:\Users\Asus\Desktop\Musa\_All\Genel Eğitim Dokümanları\Yapay Zeka\Full-stack-deep-learning"

$ git pull https://github.com/bayirm15/Full-stack-deep-learning.git