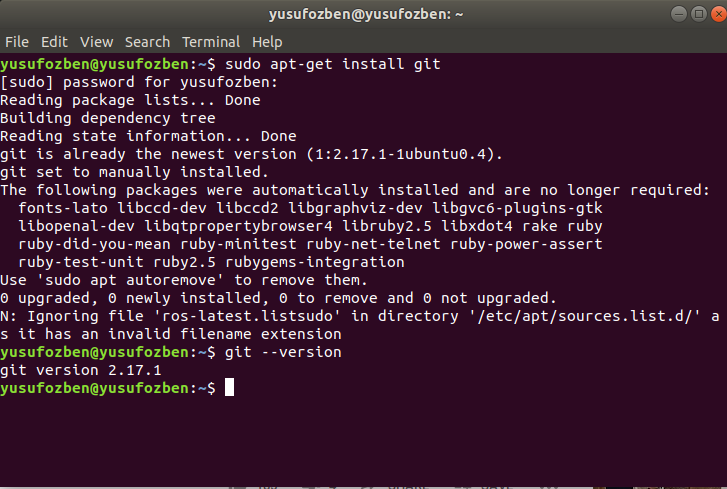
**GIT**

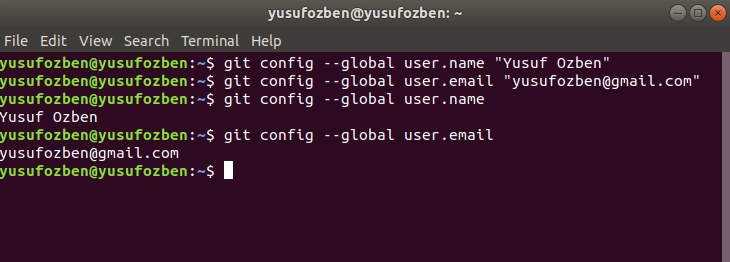
**Temel Git İşlemleri**

Git’i kullanmak için ilk olarak “sudo apt-ger install git” komutu ile git dosyalarını Ubuntu’ya indiriyoruz.

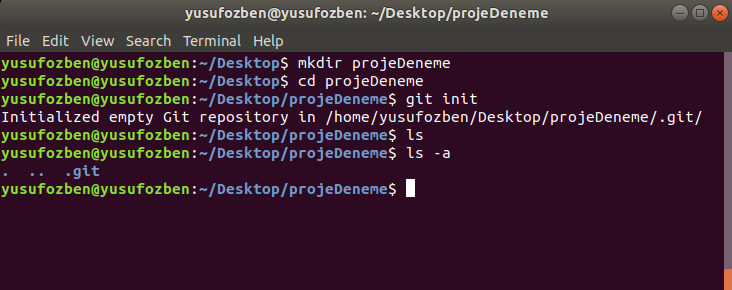
Git’in indiğini test etmek için ve ya yüklü olan versiyonunu öğrenmak için “git –version” komutunu çalıştırıyoruz.



Localde kullanacağımız Git için “git config –global user.name “Isım”” ve “git config –global user.email “email”” komutları ile konfigürasyonları gerçekleştiriyoruz. “—global” parametresi bilgisayardaki tüm dosyalar için bu konfigürasyonların geçerli olduğunu göstermektedir.



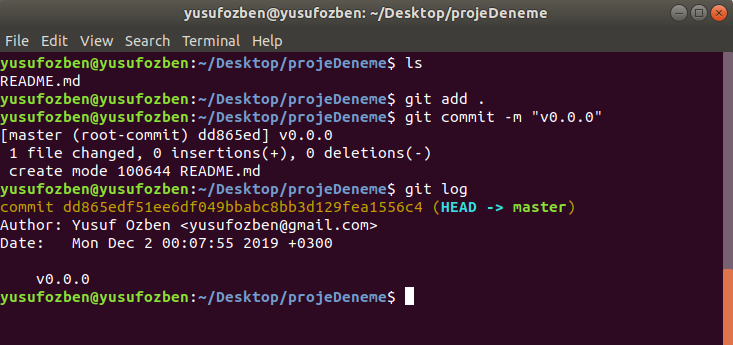
Git’e yüklemek istediğimiz dosyayı oluşturuyoruz, varsa dosyanın konumuna gidiyoruz. “git init” komutu ile dosyayı git kullanmaya hazır hale getiriyoruz. Bu komuttan sonra klasörümüzde gizli Git’in dosyaları oluşuyor. Bu dosyaları görmek için “ls –a” komutunu çalıştırabiliriz.



Git platformlarında projenin özeti ve ilk bakış için “README.md” dosyası oluşturulması tercih edilir.

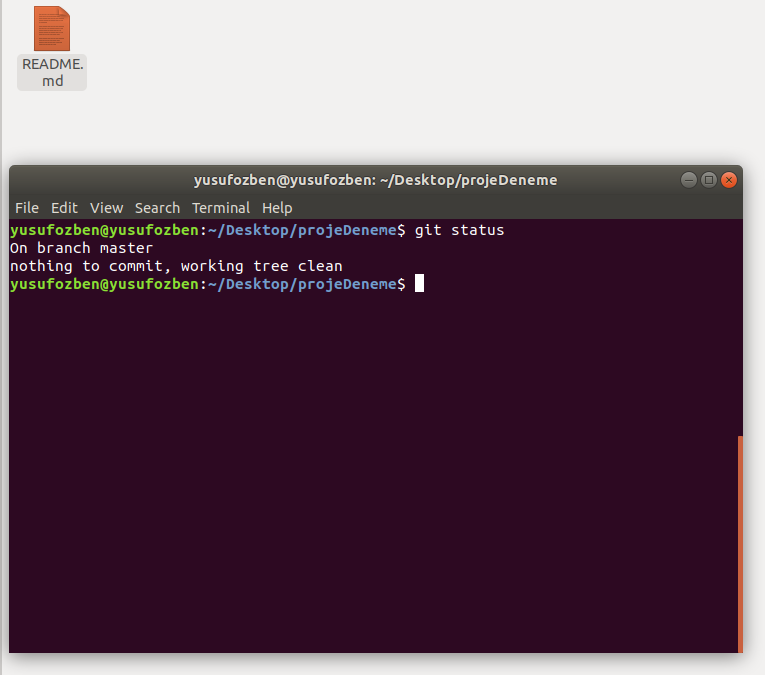
Dosyada değişiklik yapıldıktan sonra Git’e atılmak istendiğinde ilk olarak “git add .” komutu çalıştırılır. “.” Tüm dosyaların ekleneceğini ifade eder.

Sonrasında “git commit –m “versiyon adı”” komutu ile local Git’imize projemizi göndeririz. “-m” mesajı ifade eder. Commit işlemlerinde mesaj olarak versiyon adı ve ya tarih girmeyi tercih edebiliriz.

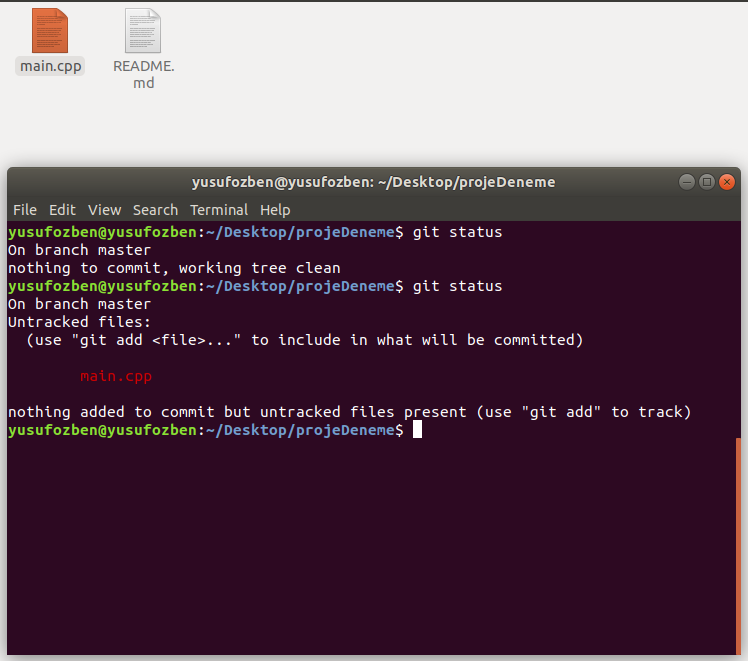


“git status” komutu ile projedeki dosyalarda commit sürecine kadar olan işlemlerde bekleyen değişiklikleri gözlemliyoruz.

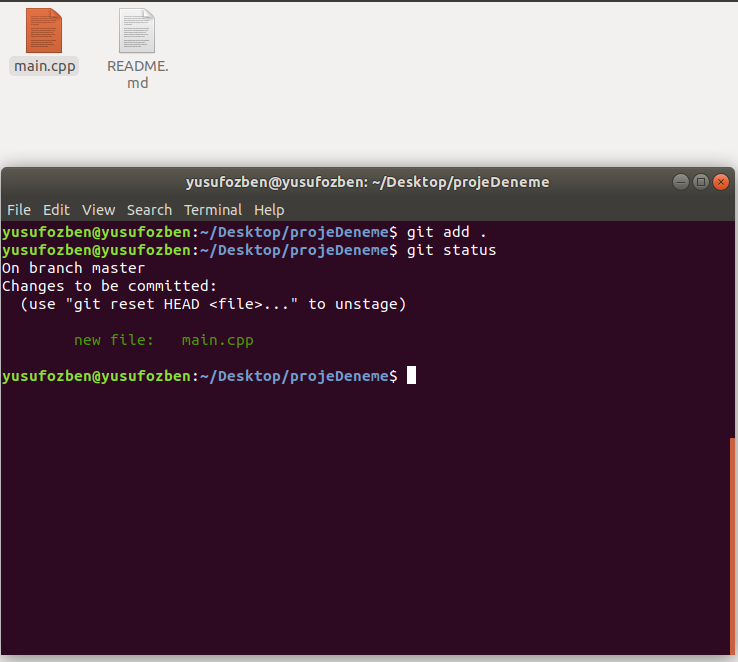
Hiçbir değişiklik yapmadığımız zaman aşağıdaki gibi bir sonuç alıyoruz.



Eğer değişiklik yapılmışsa ve “git add .” komutu çalıştırılmamışsa aşağıdaki gibi değişiklikler kırmızı ile gösterilir.

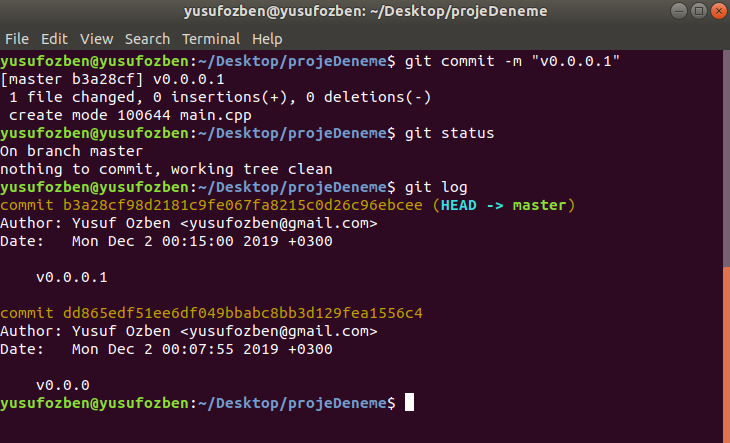


Eğer değişiklikler yapılmışsa ve “git add .” komutu çalıştırılmışsa değişiklikler yeşil olarak gözükür ve projemiz commit etmeye hazırdır.

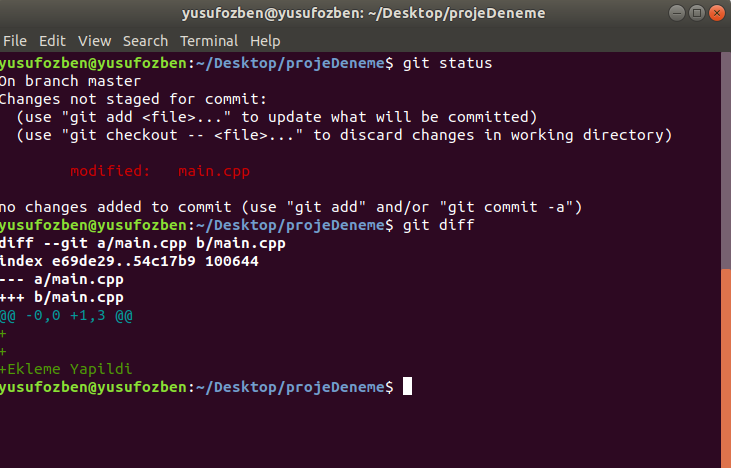


Commit işleminden sonra status comutunu çalıştırdığımız zaman yine değişiklik olmadığını bizlere göstermektedir.

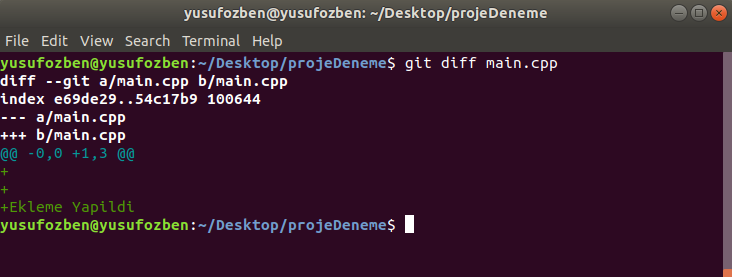
“git log” komutu ile tüm commit işlemlerimizi ve ayrıntılarını görebiliriz.



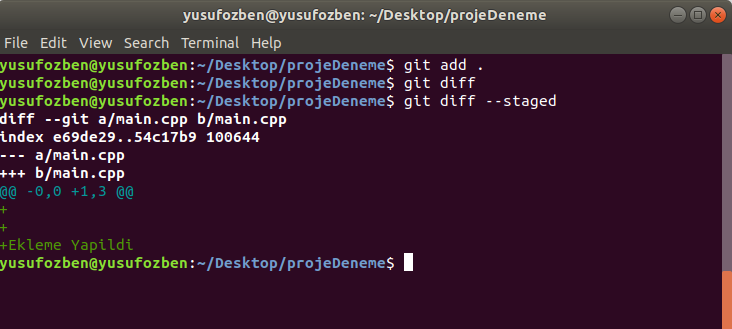
Dosya içinde yapılan değişikliklerin ayrıntılarını görmek için “git diff” komutunu kullanıyoruz. Yapılan ekleme ve çıkarmalar ayrıntılı olarak gösterilir. (“git add .” yapılmamış)



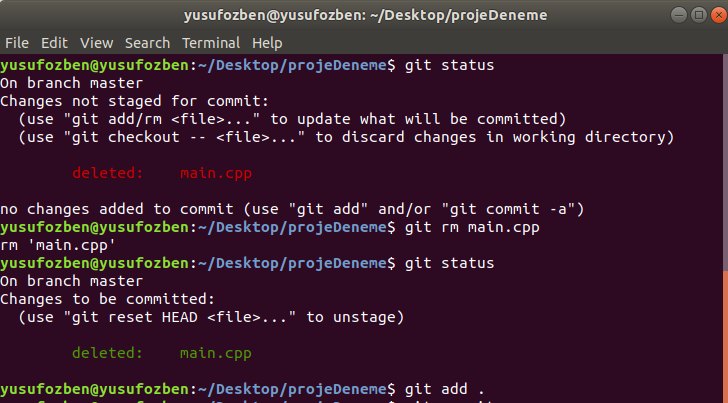
Belli bir dosyadaki değişiklikleri görmek için “git diff [dosya\_adi]” komutumuza dosya adı parametresini giriyoruz.



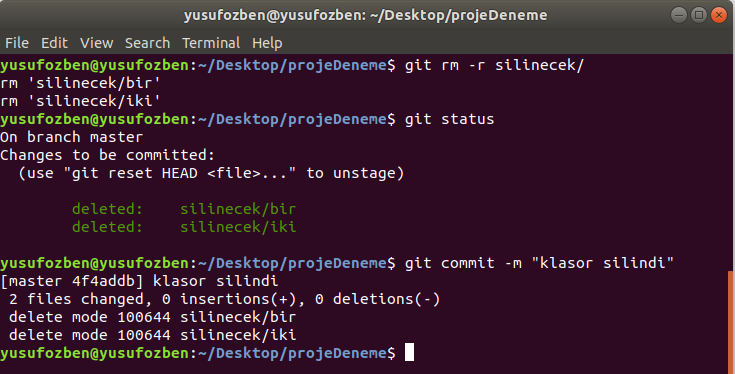
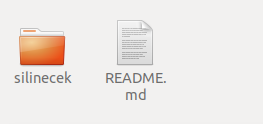
Eğer “git add .” işlemi yapılmış ve biz add-commit işlemleri arasındaki farkları incelemek istiyorsak “git diff –staged” komutumuza –staged parametresini ekliyoruz.



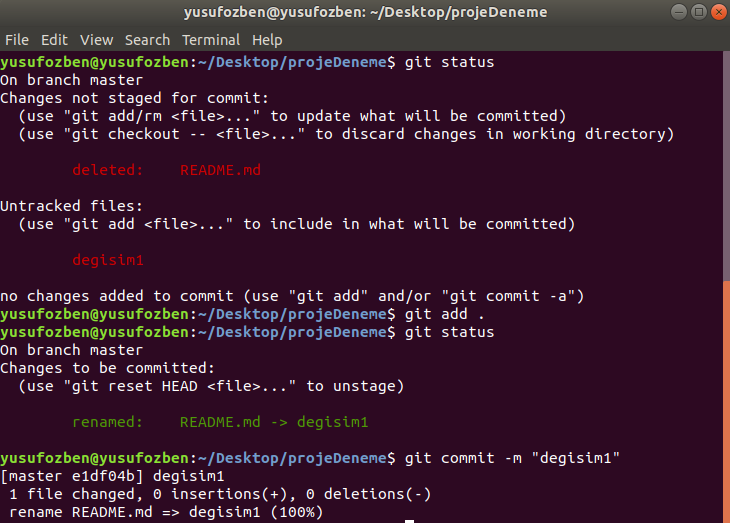
Bir dosyayı elle sildikten sonra ister “git rm [silinen\_dosya]” ister “git add [silinen\_dosya] komutu ile dosyayı commit etmeye hazır hale getirebiliriz.



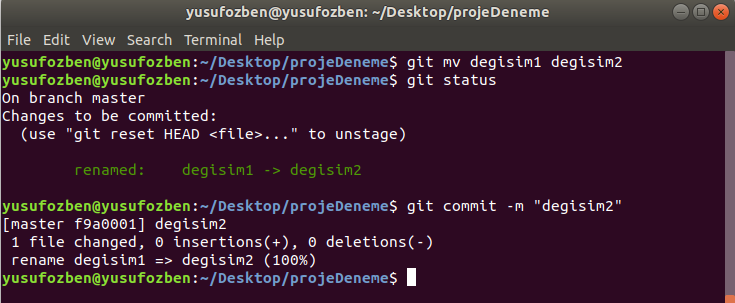
Projemizde içi dolu “silinecek” adında bir klasör olsun. Bu klasörü komut satırından “git rm –r silinecek/” komutu ile silebiliyoruz. Klasör adının sonuna koyduğumuz “/” sayesinde dosyanın içindeki tüm dosyaları da klasörle birlikte siliyoruz. Bu komutla silme işlemi yapıldıktan sonra proje commit edilmeye hazır hale gelmiş bulunuyor.



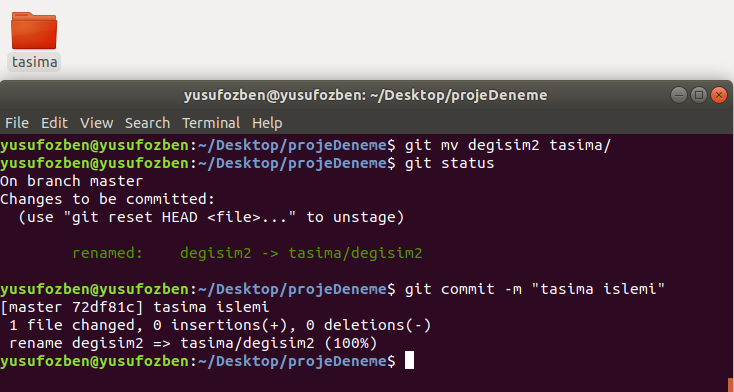
Dosya adını elle değiştirdikten sonra commit işlemine kadar aşağıdaki çıktıları alıyoruz.



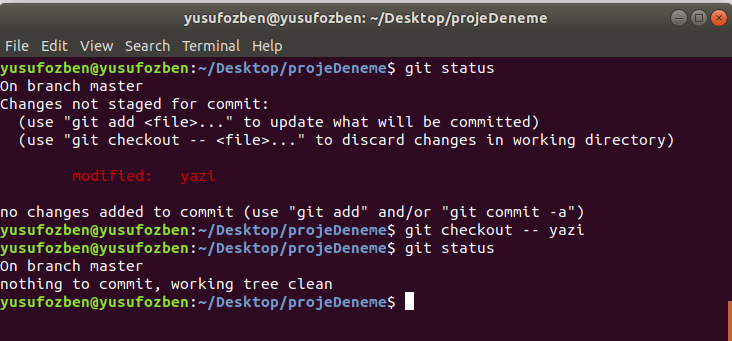
Komut satırı ile dosya ismi değiştirme işlemi “git mv eski\_isim yeni\_isim” komutu ile yapılır. Bu komutla isim değişikliği yapıldıktan sonra dosya commit edilmeye hazır hale gelmiştir.



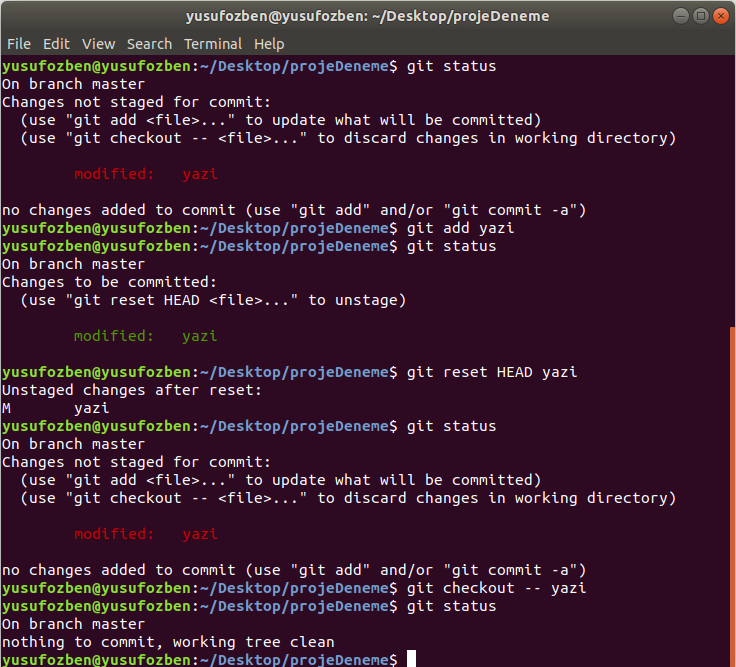
Bir dosyayı proje içinde başka bir uzantıya atmak için “git mv dosya\_adı uzantı/” komutu kullanılır. Dosya “/” tan sonraki yere konumlandırılır.



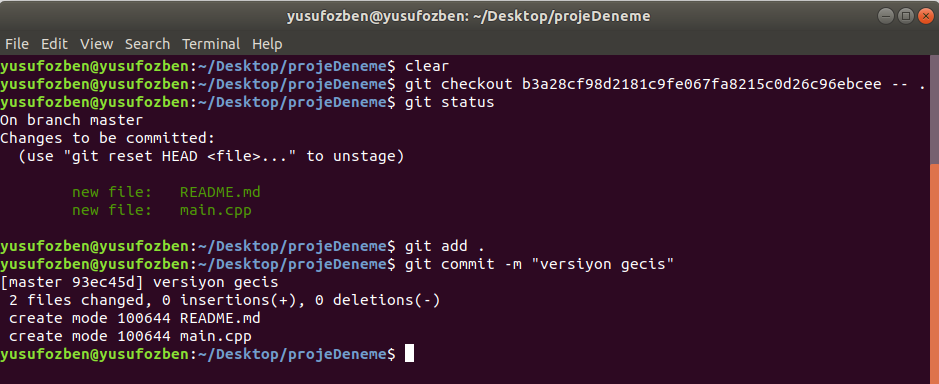
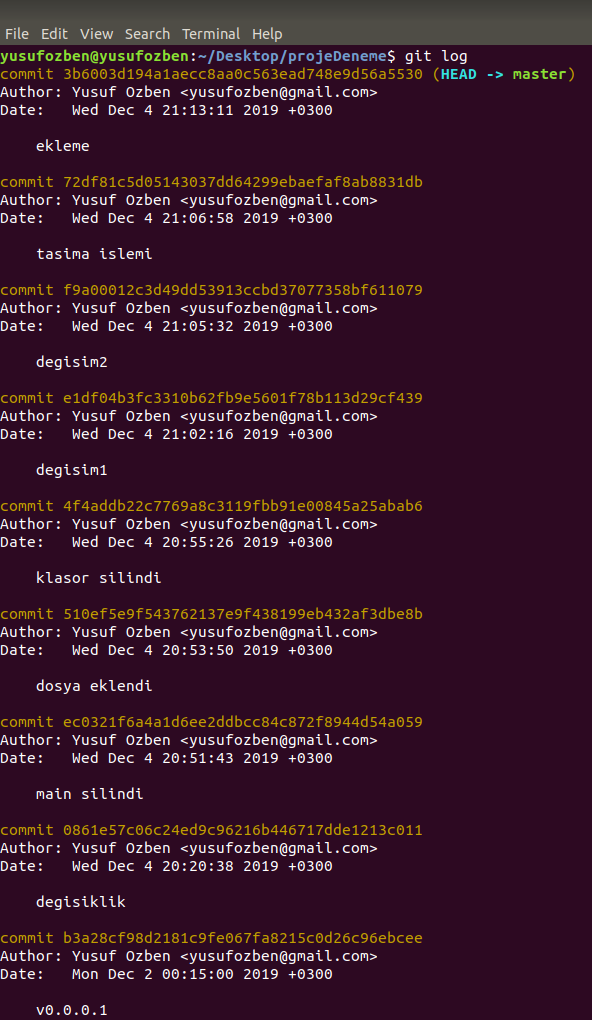
Eğer projede değişiklik yapıldıktan sonra bir önceki commit’te ki haline döndürülmek isteniyorsa “git checkout – dosya\_adı” komutu kullanılır. Girilen dosya adı bir önceki commit işlemindeki haline döner.



Eğer “git add” komutu çalıştırılmışsa ve proje en son commit haline döndürülmek isteniyorsa ilk “git reset HEAD dosya\_adi” sonrasında ise “git checkout – dosya\_adı” komutu girilerek işlem tamamlanır.

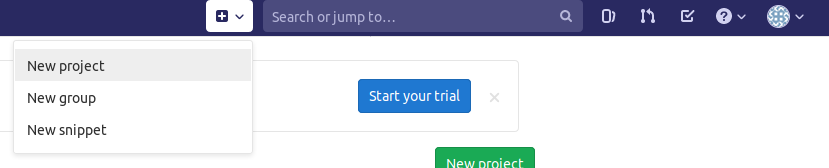


Eğer herhangi bir proje versiyonuna dönüş yapılmak isteniyorsa ilk olarak “git log” komutu çalıştırılır. Sonrasında geriye dönmek istediğimiz versiyonun kodu kopyalanır ve “git checkout versiyon\_kodu -- .” komutunun içine yazılır. Bu komutun sonuna tam geçiş yapmak “.” koymak önemlidir, yoksa versiyon geçişinde sıkıntı yaşanabilir. Geçiş işlemi yapıldıktan sonra projemize add ve commit işlemlerini uygulayabiliriz.



**GitLab**

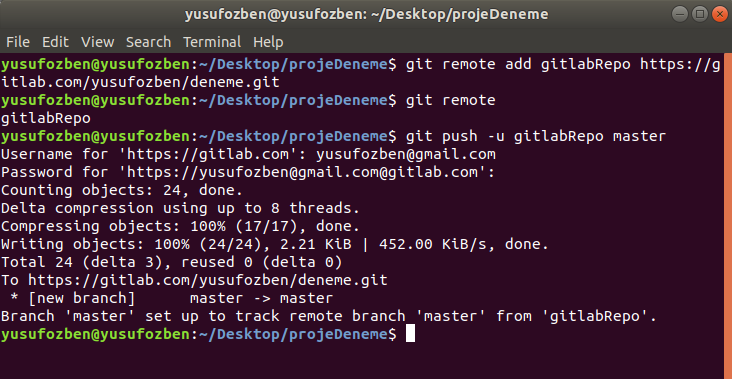
GitLab’da proje geliştirmek için ilk olarak “New Project” kısmından yeni proje oluşturuyoruz.



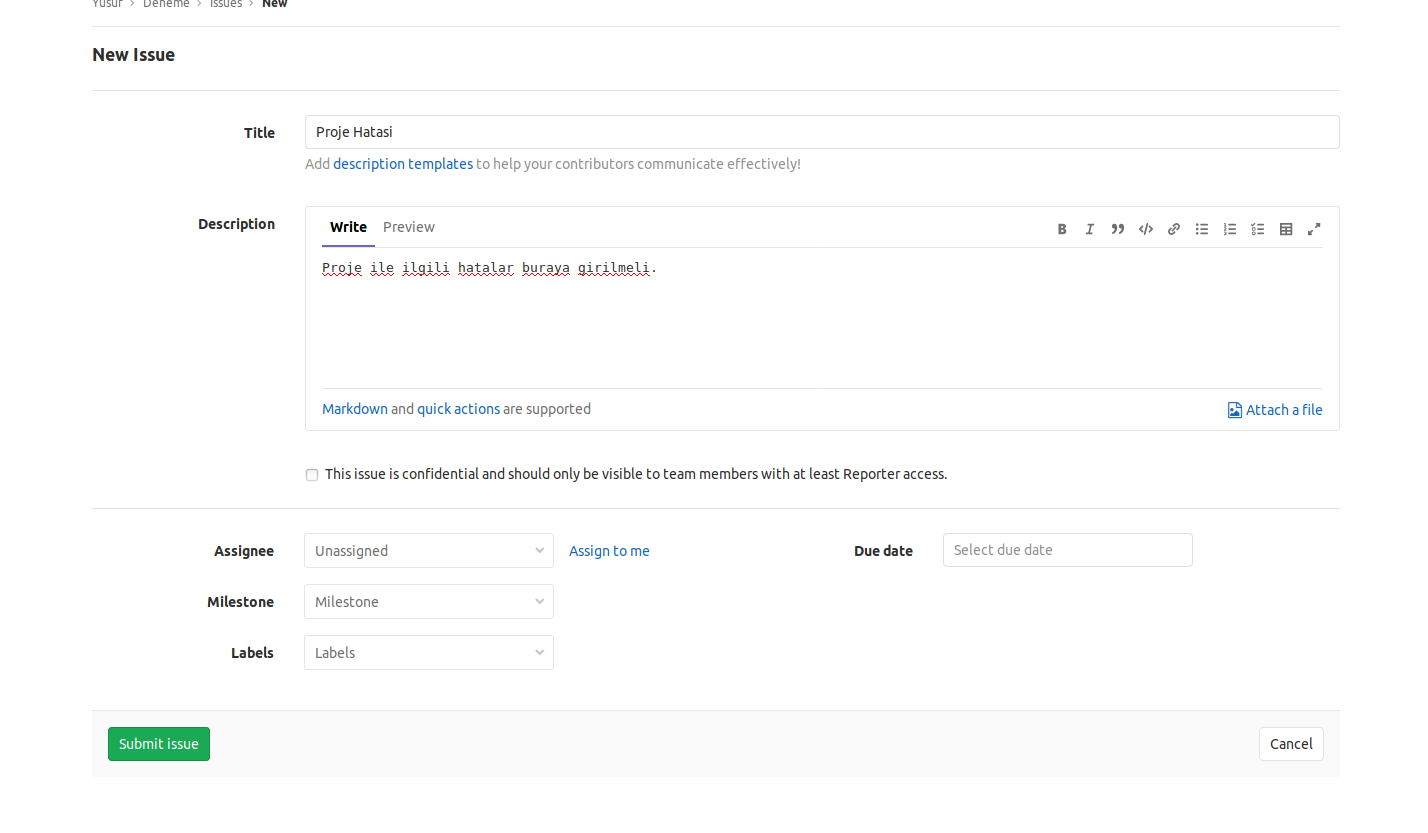
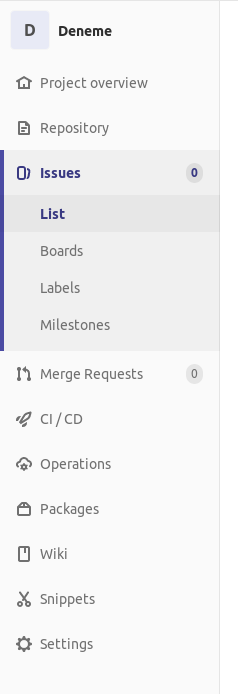
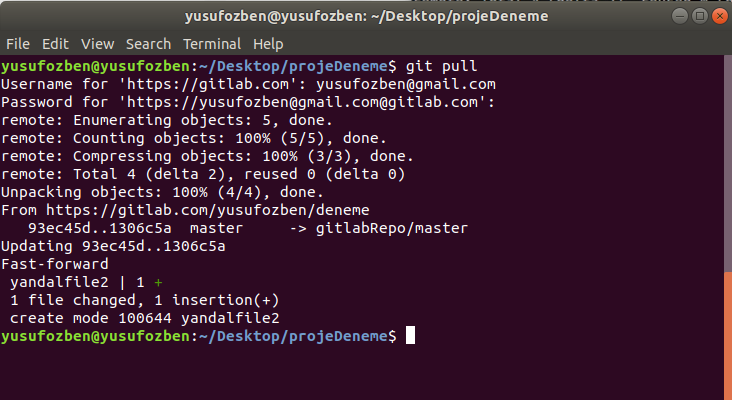
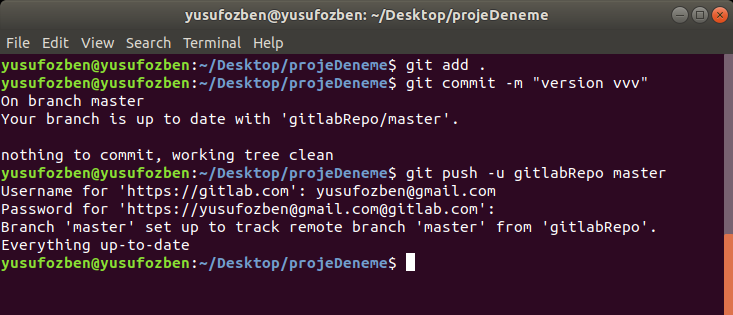
Proje oluşturduğumuz zaman gerekli ve en çok kullanılan dört git komutları proje ilk oluşturulduğu zaman GitLab tarafından verilir.



GitLab’da açtığımız projeyi bilgisayarımızda oluşturduğumuz eski git dosyasına bağlamak için “git remote add gitlabRepo [html\_address]” komutunu giriyoruz. Bu komutta girdiğimiz “gitlabRepo” takma addır, istediğimiz adı bilgisayarda projemize verebiliriz. “git remote” komutu ile git dosyamızı bağladığımız takma adı görürüz. “git push –u gitlabRepo master” komutu ile dosyada önceden gerçekleştirdiğimiz tüm commit işlemleri GitLab projemize eklendi.

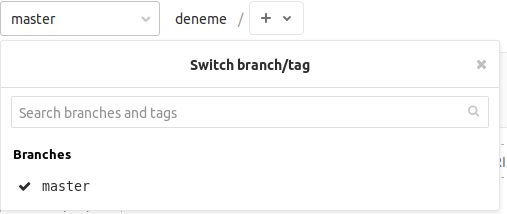


GitLab’da Git’e ek olarak her commit işleminden sonra “git push –u [takma\_ad] [branch]” komutunu çalıştırcaz. Proje sırasında genel olarak master branch’ini kullanacağız.

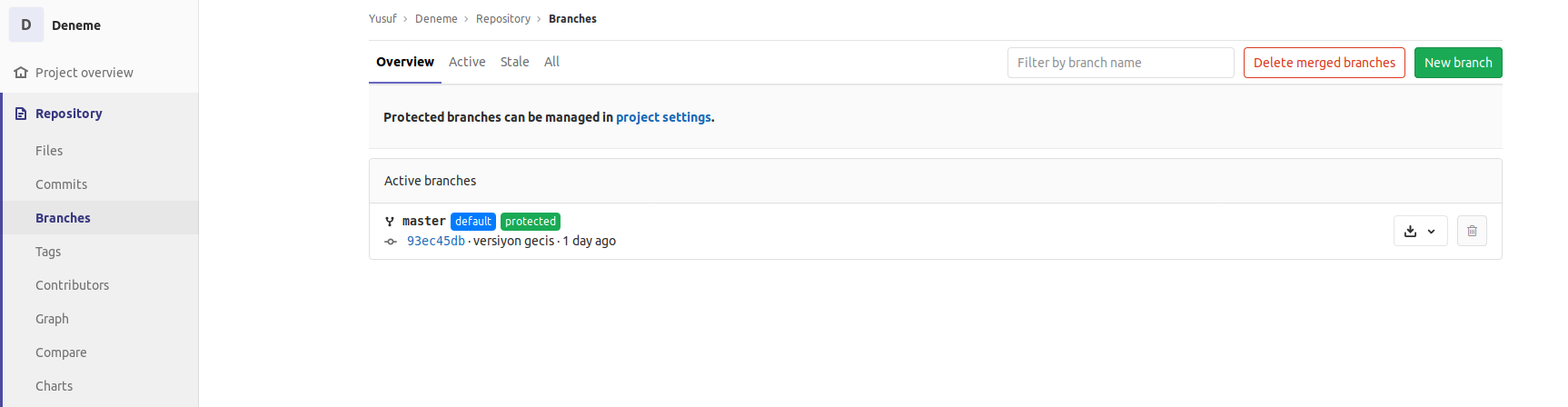


**Branch Kullanımı**

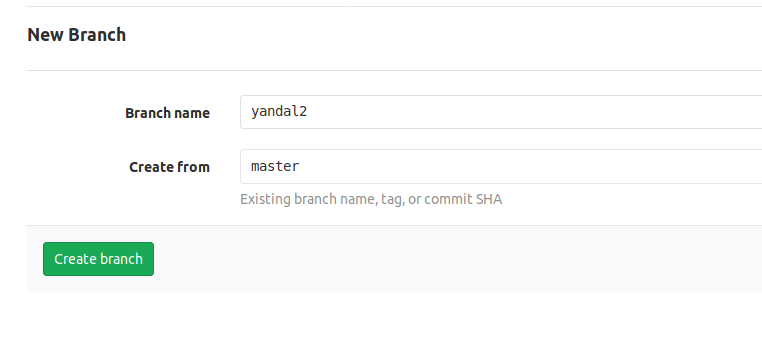
Branch projemizi asıl gittiğimiz geliştirme yolunda dallandırmamızı ve uygun gördüğümüz zaman asıl projemizle birleştirdiğimiz (merge) yoldur. GitLab projemizde aşağıdaki gibi “+” kısmına basarsak mevcut branch’lerimizi görürüz.



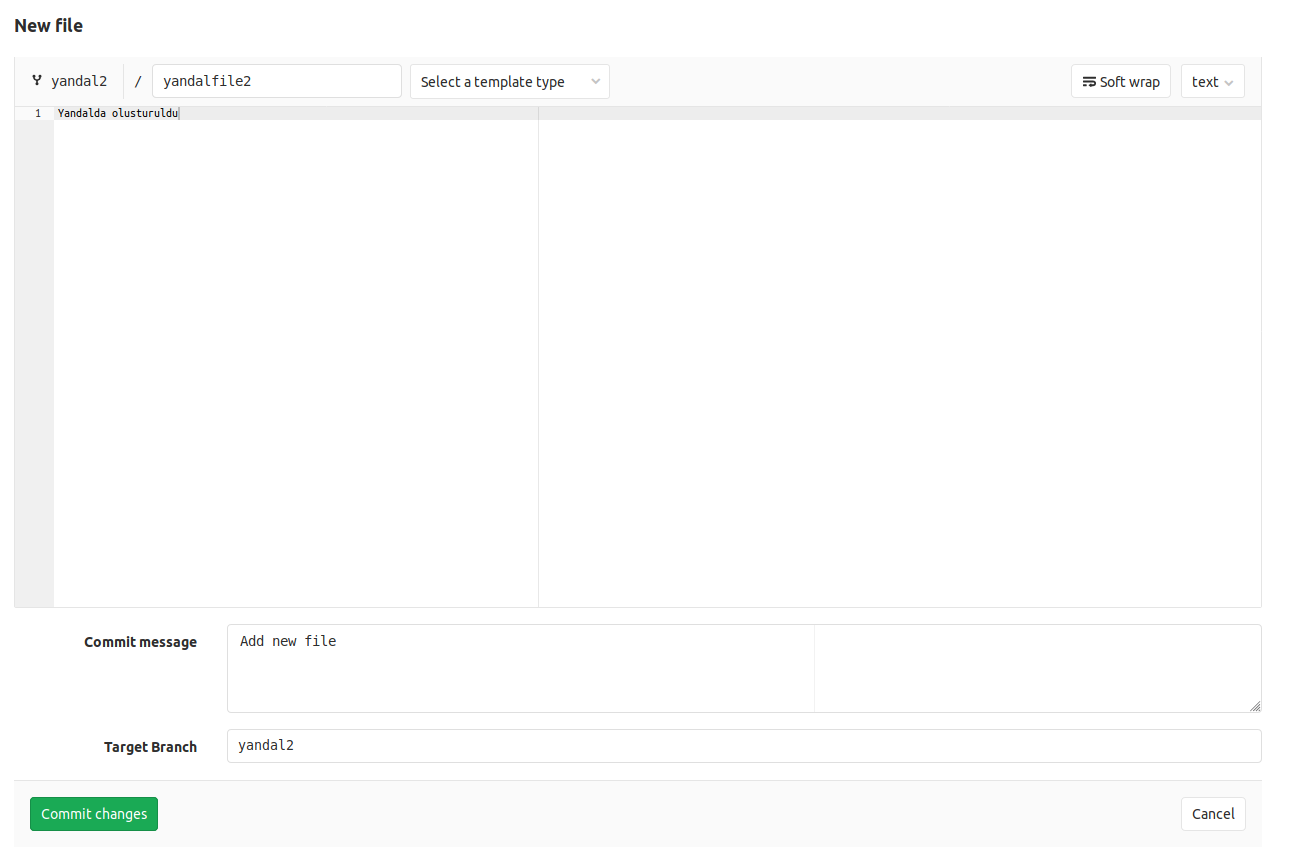
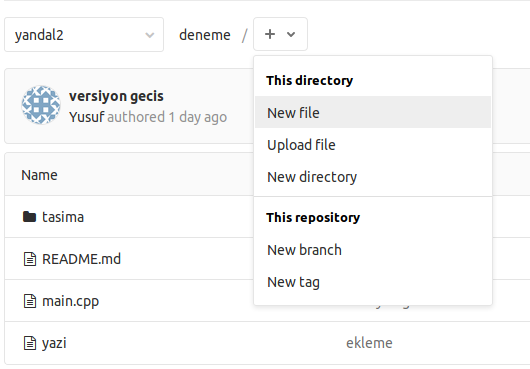
Yeni branch oluşturmak için soldaki menüden “Branches” a basıyoruz. Sonrasında sağ üstte bulunan “New Branch” e basıyoruz.



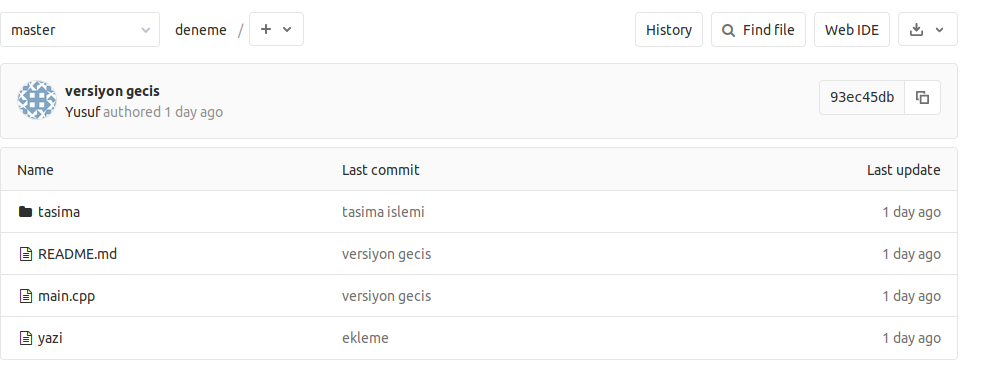
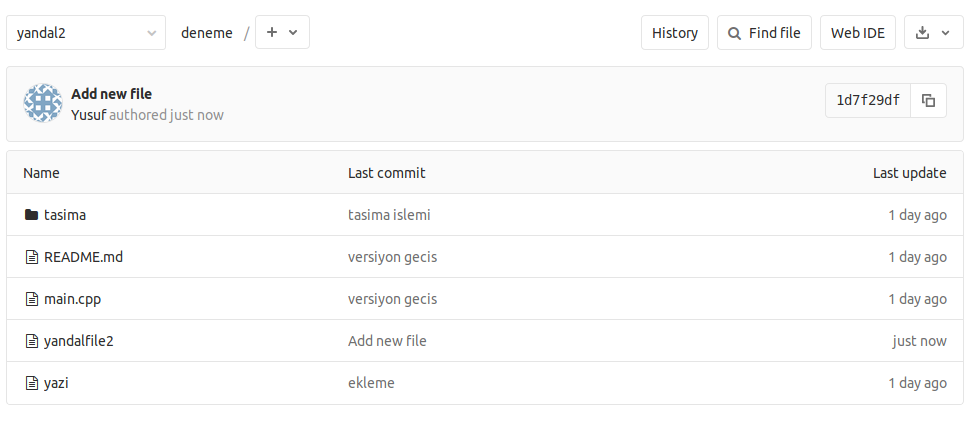
Branch’imizin adını ve hangi branch’den oluşturulacağını seçiyoruz.



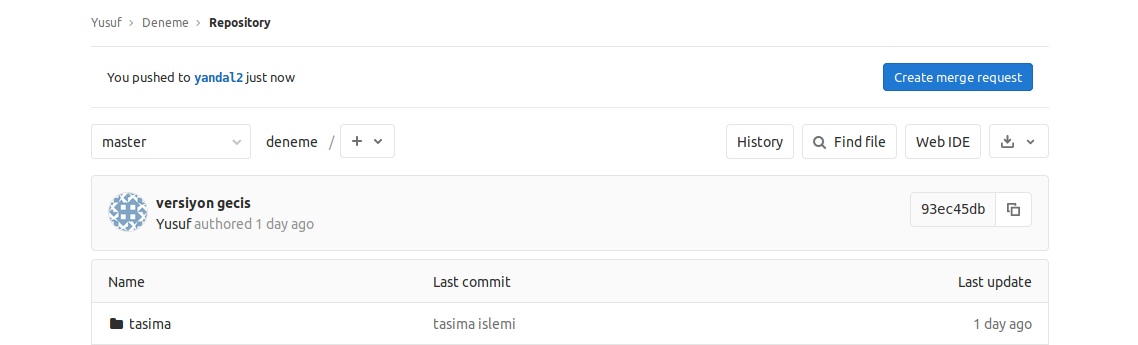
Merge işlemini denemek için “yandal2” içinde “yandal2” adında bir dosya oluşturuyoruz ve dosyanın içine bir şeyler yazıyoruz.



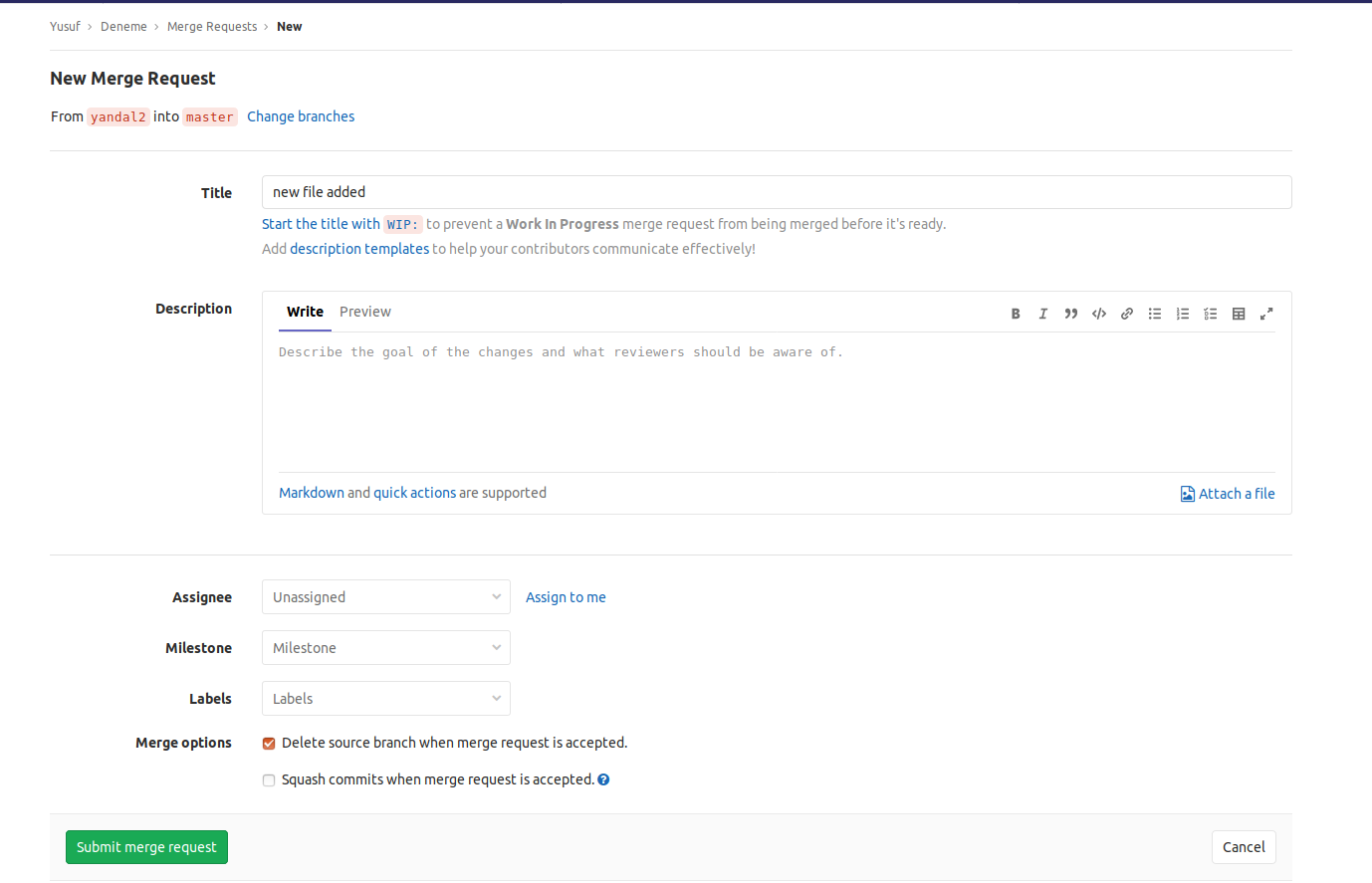
Proje dosyalarında Branch’imizi değiştirerek yaptığımız değişikliği ve iki branch’in birbirlerini etkilemediğini görebiliriz.



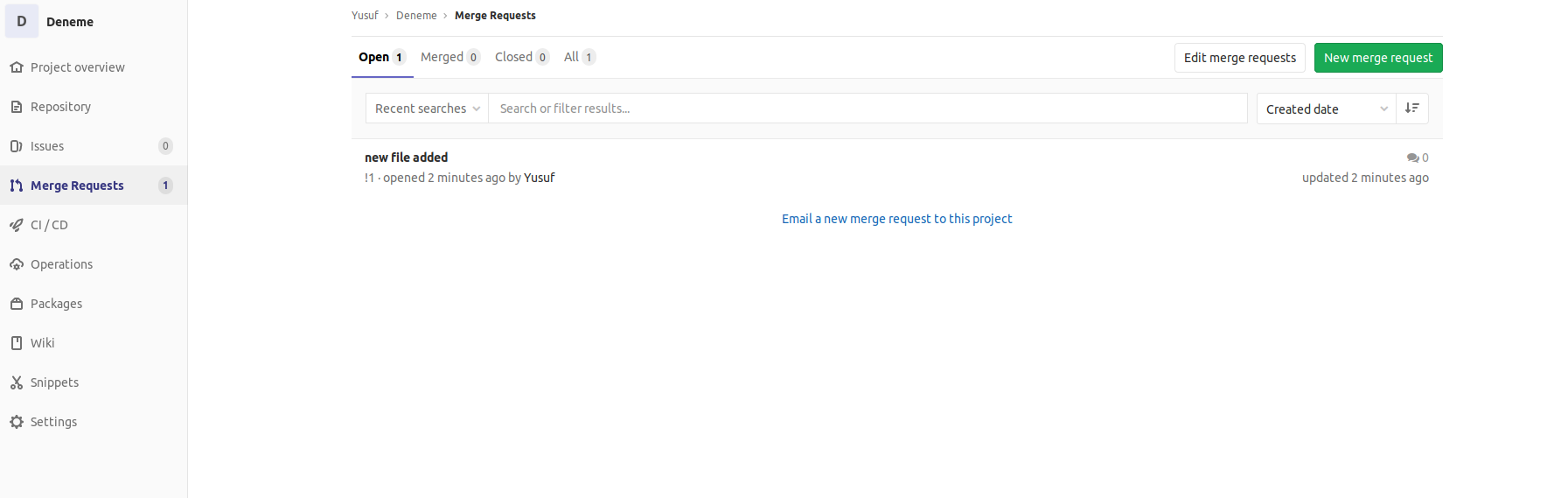
Bu iki Branch’imizi birleştirmek için “Repository” bölümündeki “Create merge request” butonuna basıyoruz.



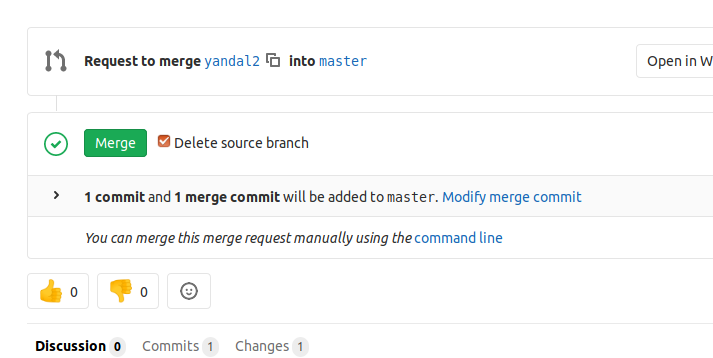
Aşağıdaki gibi gerekli talep özelliklerini dolduruyoruz.



Menüdeki “Merge Request” başlığna gidip “Merge Request” leri görebiliriz.



Merge talebine girdikten sonra “merge” butonuna tıklayarak merge işlemini tamamlayabiliriz.



Bu işlemden sonra görüldüğü gibi “yandal2” Branch’inde yaptığımız değişiklikler “master” Branch’ine geçmiş oldu.

