**Remote Repository'ler**

Günlük çalışmamız sırasında staging ve commit gibi versiyon kontrolü ile ilgili işlemlerin çoğunu yerel diskimizde yer alan local repository üzerinde yaparız. Proje'de çalışan tek kişi siz iseniz muhtemelen Internet'de veya yerel ağıda yer alan remote bi repository oluşturmanıza da gerek olmayacaktır.

Ancak takım çalışması söz konusu olduğunda, takımdaki geliştiricilerin birlikte çalışabilmesi için herkesin değişikliklerini ortak bir alanda yayınlaması ve diğerlerinin de bu ortak alan üzerinden bu değişiklikleri kendi branch'lerine entegre etmesi gerekecektir. Bu durumda başvuracağınız en etkin araç Git'deki Remote Repository işlevleridir. Remote repository'leri en basit anlamda tüm ekibin erişimi olan dosya sunucusu olarak düşünebilirsiniz.

Gelin şimdi Local ve Remote repository'leri birbirinden ayıran temel özelliklere göz atalım

**Konum**

Local repository'ler geliştiricilerin kendi bilgisayarlarında yer alırken Remote repository'ler, çoğunlukla internet olmak üzere, ekipteki herkesin erişebileceği bir sunucuda yer alırlar.

**Özellikler**

Teknik olarak remote repository'ler ile local repositoryler arasında bir fark yoktur. Local repository'ler için önceki bölümlerde ele aldığımız commit işlemi, branch oluşturma gibi işlemlerin tamamı remote repository'ler için de yapılabiliyor. Ancak tüm bu benzerliklere rağmen remote repository'ler için Working Copy (aktif branch'deki dosyaların diskimizdeki kopyaları) yapısı geçerli değildir, remote repository'lerde sadece Git'in veri tabanının tutulduğu **.git** klasörü yer alır.

**Repository Oluşturma**

Local bir repository ancak iki şekilde oluşturulabilir

* Boş bir repository olarak sıfırdan **git init** komutu ile oluşturabilirsiniz veya
* Remote bir repository'yi **git clone** komutu ile yerel diskinizde indirebilirsiniz.

Remote repository'ler de iki yöntem ile oluşturulabilir

* Local repository'nizi **git clone** komutunu **--bare** parametresi ile kullanarak remote bir repoository'ye klonlayabilirsiniz veya
* Boş bir remote repository oluşturmak için **git init** komutunu yine **--bare** parametresi ile kullanabilirsiniz.

**Local/Remote iş akışı**

Git'de remote repository işlemleri için az sayıda komut vardır. Günlük çalışmamız srasında bölümün başında da belirttiğimiz gibi Git işlemlerimizin çoğu local repositorymiz üzerinde gerçekleşir ve internet veya ağ bağlantısına ihtiyaç duymayız. Ancak remote repository komutlarını kullanabilmek için internet veya ağ bağlantısına ihtiyaç vardır.

Bu bölümümüzde Remote Respository'ler ile ilgili aşağıdaki konuları ele alarak ayrıntıları öğreneceğiz

* Remote Bir Repository'ye Nasıl Bağlantı Sağlanır
* Remote Repository'deki Verilerin İncelenmesi
* Remote Değişiklikleri Entegre Etmek
* Local Bir Branch'i Yayınlamak (Publish)
* Branch'leri Silmek

# Remote Bir Repository'ye Bağlantı Sağlamak

Remote bir repository'yi yerel diskinize **git clone** komutu ile indirdiğinizde Git otomatik olarak bu işlemi yapmak için kullandığınız bağlantı bilgilerini hatırlar. Git bu bilgi'yi varsayılan olarak **origin** adı verilen remote bir repository olarak kayıt altına alır. Local olan bir respository için ise böyle bir bilgi tutulmaz. Ancak bölüm girişinde de ele aldığımız gibi Local bir repository'yi baz alarak yeni bir remote repository oluşturabiliriz. Bunun için **git clone** komutunu kullanabiliriz. Örneğin



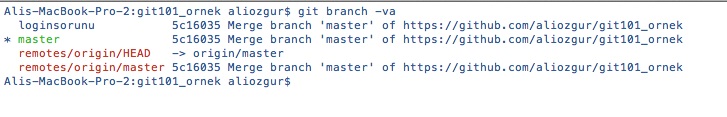
Yukarıdaki ekran görüntüsünde ilk komutumuz olan **git remote add**ile local repository'miz ile remote repository'miz arasındak bağlantıyı kuruyoruz. İkinci komutumuz olan **git remote -v** ile de remote repositorymiz ile ilgili bilgileri görebiliriz.

Dikkat ettiyseniz her bir remote repository için biri **fetch** diğeri de **push** işlemleri için kullanılan iki adres bulunur. **fetch** adresini remote repository'den yapılacak olan okuma işlemleri, **push** adresini de remote repository'ye yapılan yazma işlemleri için kullanılır. Genel olarak bu iki adres aynı olmakla birlikte performans ve güvenlik gibi gerekçeler ile iki farklı adres de kullanılabilir.

Local bir repository'nizi istediğiniz sayıda remote repository ile ilişkilendirebilirsiniz. Yukarıdaki ekran çıktısında sadece bizim oluşturduğumuz **git101\_ornek** isimli remote listeleniyor, birden fazla remote ilişkisi olsaydı hepsi listelenecekti.

**Remote Repository'deki Verilerin İncelenmesi**

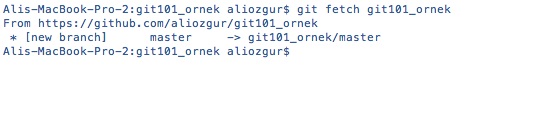
**git clone** komutu remote bir repository'yi yerel diskimize indirdikten sonra **git branch -va** komutunu çalıştırdığımızda aşağıdaki görüntüde yer alan bilgiler listelenecektir.



Dikkat edecek olursanız local repository'lerimiz hala yerinde duruyor ancak listemizde ilave olarak **origin/HEAD** ve **origin/master** isimli iki remote kaydı var. Peki daha önceki bölümde **git add git101\_ornek**komutu ile oluşturduğumuz remote repository kayıtlarımız neden listlenmiyor? Bunun nedeni önceki bölümde kullandığımız **git add** komutu ile local ve remote repository arasında sadece bir ilişki/bağlantı tanımladık, aslında bu komut sonrasında local ve remote arasında herhangi bir veri trasferi gerçekleşmez.

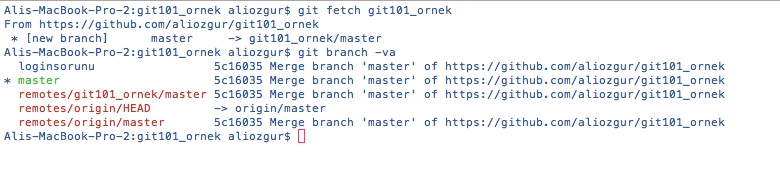
**Remote Repository bilgileri güncel olmayabilir!** Git remote repository'ler ile ilgili yerel diskinizde bir takım bilgileri içerir. Ancak Git arka planda otomatik olarak bu bilgileri sizin için belirli aralıklarda güncellemez! Bu işlemin gerçekleşmesi ve sizin diğer takım arkadaşlarınızın yaptığı değişikliklerden haberdar olabilmeniz için Git'e bu bilgileri güncellemesini söylemeniz gerekir.

Git'in remote repository ile ilgili yerel diskinizde tuttuğu bilgileri güncellemesini sağlamak için **git fetch** komutunu kullanmanız gerekir.

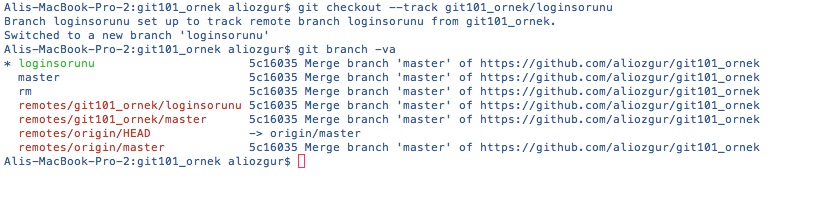


Fetch komutu yerel diskinizdeki branchlerinizi ve Working Copy'deki dosyalarınızı güncellemez veya değiştirmez. Bu komut ile sadece takım arkadaşlarınızın remote repository'de yayınladıkları değişikliklere ilişkin bilgiler yerel diskinize indirilir. Daha sonra bu değişikliklerden hangilerini hangi local branch'e entegre edeceğinize kendiniz karar verebilirsiniz.

Bu işlemden sonra tekrar **git branch -va** komutunu çalıştırdığımızda **gitornek\_101/master** isimli remote repositorymizdeki branchlere ilişkin bilgileri de görebiliriz.



Bilgilerini güncellediğimiz git101\_ornek/master isimli branch'de değişiklikler yapmak için öncelikle bu branch'i baz alarak yeni bir local branch oluşturup dosyaların Working Copy alanımıza kopyalanmasını sağlamamız gerekiyor. Bunun için **git checkout** komutunu **--track** parametresi ile kullnıyoruz.



**git checkout --track** komutu ile aşağıdaki işlemler gerçekleşir

1. Remote branch ile aynı isimde local bir branch oluşturulur
2. Yeni oluşturulan branch aktif hale getirilir
3. --tracking parametresini kullandığımız için yeni oluşan local branch ile remote branch arasında "tracking relationship" adı verilen ve local branch'in hangi remote branch'deki değişiklikleri takip ettiğini gösteren ilişki kurulur

**Tracking Relationship (Takip İlişkisi)**: Git'de daha önceki bölümlerde de bahsettiğimiz gibi branchler aslında birbirinden tamamen bağımsızdır ve aralarında doğrudan bir ilişki yoktur. Ancak *track* parametresi ile local bir branch'in hangi remote branch'deki değişiklikleri takip edeceğini tanımlayabiliriz. Bu durumda Git iki branch'den herhangi birinde yer alan ancak diğerinde yer almayan commit'leri tespit ederek bizi bilgilendirecektir. Yani

* Local branch'inizde remote branch'e yayınlamadığınız (push) commit'ler varsa bu durumda local branch'inizin remote branch'den önde (ahead) olduğu
* Takım arkadaşlarınız remote branch'e bazı commitleri push ettiğinde ve siz de local branch'inizi güncellemediğiniz durumda local branch'inizi remote branch'in gerisinde (behind) olduğu bilgisi Git tarafından "Tracking Relationship" tanımı sayesinde **git status** komutunun çıktısı olarak gösterilir

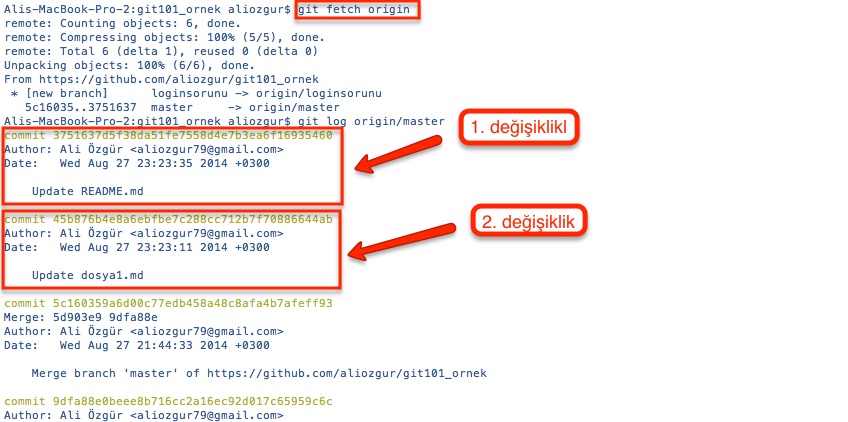
Local branch'imizi hazırladığımıza göre gelin şimdi birkaç değişiklik yapalım. Bu değişiklikleri yaptıktan sonra her zamanki gibi önce değişikliklerimizi Staging Area'ya alıyoruz ve sonrasında da commit işlemini gerçekleştirerek local repository'de versiyon kontrolüne ilişkin işlemlerimizi bitiriyoruz. Son adım olarak da **git push** komutu ile localdeki bu değişikliklerimizi remote branch'de yayınlıyoruz.



**git push** push komutu aslında **git push**formatındadır. Ancak local branch'imizi oluştururken kullandığımız *track* parametresi sayesinde kurulan "Takip İlişkisi" sayesinde push komutunun uzun hali yerine sade hali olan **git push** formatında kullanabiliyoruz.

**Remote Değişiklikleri Entegre Etmek**

Takım arkadaşlarınız kendi değişikliklerini tamamlayıp remote branch'de yayınladıktan sonra siz de bu değişiklikleri inceleyip kendi local branch'inize entegre ederek çalışmanıza devam edebilirsiniz. Ancak remote branch'deki değişiklikleri entegre etmeden önce bu değişikliklere ilişkin bilgileri (dosyaları değil sadece değişikliklere dair Git'de tutulan bilgiler) görmeniz ve incelemeniz gerekir.



Remote branch'deki değişiklikleri indirmek için **git fetch** komutunu kullanıyoruz. Git fetch komutuna geçilen *origin* değeri ise daha önceki bölümlerde gösterdiğimiz *remotes/origin/master* isimli remote branch bağlantısına referans vermek için kullanılır.

*origin* değeri **git fetch** komutunun bir parçası değil sadece bir parametre. Origin yerine daha önce local branchimiz ile bağlantısını/ilişkisini kurduğumuz herhangi bir remote branch'i gösteren bir değer olabilir.

*git fetch* komutu ile remote branch'deki değişiklikleri indirdikten sonra ise **git log** komutunu kullanarak bu remote branch'deki değişiklikler ile ilgili bilgileri görebiliriz. (değişiklik tarihi, kimin yaptığı, değişen dosyalar ve commiti sırasında girilen mesaj gibi)

Değişiklikleri inceledikten sonra bunları local branch'inize entegre etmeye karar verdiğimizde ise **git pull** komutunu kullanmamız gerekecek

Remote branchdeki değişikliklerin bilgilerini indirmek için kullanılan **fetch** (türkçe anlamı [getirmek](http://www.seslisozluk.net/?word=fetch&lang=tr-en)) ve bu değişiklikleri entegre etmek için kullanılan **pull** (türkçe anlamı [çekmek](http://www.seslisozluk.net/?word=pull&lang=tr-en)) ifadelerinin birbirine yakın anlamları olduğu için karıştırabilirsiniz. Bu karışıklığın önüne geçmek için yapacağınız en güzel şey **git pull** komutunu hiç kullanmamak olacaktır. Ayrıntılar için İngilizce bir blog post olan [*Git: fetch and merge, don't pull*](http://longair.net/blog/2009/04/16/git-fetch-and-merge/) inceleyebilirsiniz.

Git pull komutu aslında arka arkaya iki şey yapmanızı sağlar

* Remote branch'deki değişiklikler ile ilgili bilgileri indirmek, yani **git fetch**
* Remote branch'deki değişiklikleri local branch'inize entegre etmek yani **git merge**

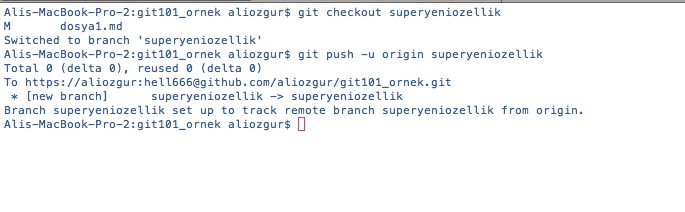
İlerleyen bölümlerde çakışmaların tespit edilmesi, çözülmesi ve değişikliklerin entegre edilmesi konularını ayrıntılı olarak ele alacağız şimdilik sadece iş akışımızı özetleyip bu konuyu burada sonlandıralım. Akışımız özetle şöyle olacak

* git fetch : remote'dan güncelleme bilgilerini indir
* git diff : remote ve local arasındaki farkları incele
* git merge : değişiklikleri otomatik merge et çakışma varsa bir sonraki adıma geçin
* Çakışma olan dosyalarınızı açın ve çakışmaları düzeltin
* git add: çakışmanın giderildi ve değişiiklik Staging Area'ya alındı

# Local Bir Branch'i Yayınlamak (Publish)

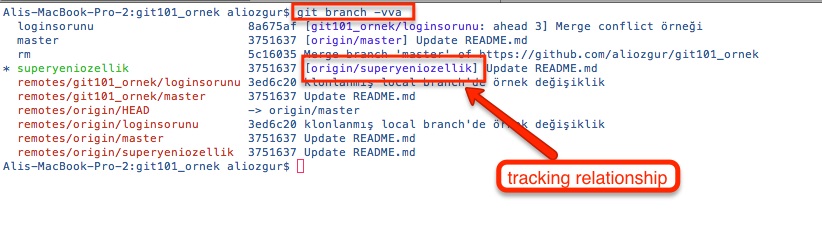
Kendi bilgisayarınızda oluşturduğunuz Local bir branch siz yayınlamaya karar vermediğiniz sürece sadece sizin bilgisayarınızda yer alacaktır. Yani local bazı branchlerinizi sadece kendi bilgisayarınızda tutarken istediklerinizi de takım arkadaşlarınız ve hatta tüm dünya ile paylaşabilirsiniz.

Gelin şimdi **superyeniozellik** isimli local branch'i remote repositorymizde paylaşalım.



Önce **git checkout** komutu ile branch'imizi aktif hale getiriyoruz ve sonra **git push** komutu ve **-u** seçeneği ile local branch'imizi remote repository'mizde yayınlıyoruz. Push komutu için verdiğimiz origin ve superyeniozellik değerleri ile **HEAD** branch'imizi **origin** remote repository'de **superyeniozellik** isimli branch olarak yayınlanmasını istediğimizi tanımlıyoruz. -u seçeneği ise local branchimiz ile remote branchimiz arasında, önceki bölümlerde de bahsettiğimiz, Takip İlişkisi (Tracking Relationship) kurulmasını sağlar.

git branch komutunu **-vva** seçeneği ile çalıştırdığınızda kurulmuş Takip İlişkisi bilgilerini de görebilirsiniz.

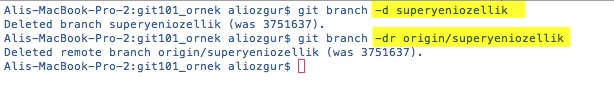


Local branch'i remote repository'de yayınladıktan sonra local branch'de yaptığımız değişiklikleri **git push** komutunu parametresiz kullanarak remote branch'imizde yayınlayabiliriz.

Artık remote repository'ye erişim yetkisi olan herkes **superyeniozellik** isimli bu branchinizi görebilir ve bu branch'i baz alarak kendi değişiklikleri üzerinde çalışma yapabilir.

# Branch'leri Silmek

Bir önceki bölümde oluşturduğumuz **superyeniozellik** isimli branch üzerindeki çalışmamızı tamamlayıp kalite kontrol sürecimizi de işlettikten sonra bu değişiklikleri **master** branch'imize entegre ettiğimizi varsayalım. Bu entegrasyon sonrasında **superyeniozellik** isimli branch'e ihtiyacımız yok ve artık bu branch'i silebiliriz. Bu branch'i kendi bilgisayarımızdan silmek için **git branch -d superyeniozellik** komutunu, remote repository'den silmek için de **git branch -dr superyeniozellik** komutunu kullanabiliriz.



Silmek istediğiniz local branch aktif ise **git branch -d** komutu hata verecektir. Silme işlemi öncesinde sileceğiniz local branch'den farklı bir branch'i **git checkout** komutu ile aktif hale getirmeyi unutmayın.

Remote branch'i **git branch -dr** komutu ile sildiğiniz halde remote repository'ye erişip branchleri kontrol ederseniz **superyeniozellik** isimli branch'in sunucuda hala durduğunuz göreceksiniz. Bunun nedeni **git branch -dr** komutundaki seçeneklerden **r** seçeneğinin sunucudaki branch'i değil yerel bilgisayarınızda remote branch bilgilerini siler. Bu değişikliğin sunucuda da geçerli olması için yani sunucudaki branch'i de silmek için **git push origin :superyeniozellik** komutu ile değişikliği bir anlamda remote repositry'de yayınlamanız gerekiyor.

[Daha ayrıntılı bilgi için bakınız (StackOverflow - İngilizce )](http://stackoverflow.com/questions/24216725/deleting-remote-branch-does-not-remove-from-github)