### Git & Github Başlangıç Rehberi



www.github.com

Bu yazıda projeler büyüdükçe gereksinimleri artan versiyon kontrol sistemlerinden biri olan git & Github konusunu işlemek istiyorum. Tabi bir çok versiyon kontrol sistemi mevcut ama en popüler olanı **“git”.** Bu yazı dizisinde bu alanla ilgili konularda sıfırdan başlangıç yapmak isteyenler için ve de çeşitli komutları bir arada görmek isteyenler için çok faydalı olacağına eminim.

**Git Nedir?**

Bu soruya öncelikle şöyle başlamak gerekiyor. Git Github demek değildir. Git lokalde yani bilgisayarlarımızda versiyon kontrolü yapmamızı sağlayan bir komut satır uygulaması. Bu uygulama ile projelerimizi versiyonlaşabilme özelliği katmış oluyoruz.

**Github’a gelecek olursak,**

Github ise temelinde git i kullanan web tabanlı bir ara yüz. On binlerce kullanıcı projelerini aynı zamanda buradan paylaşarak (Github serverlarına yükleyerek ve public bir şekilde erişime açarak ) dünyadaki diğer kullanıcılarla birlikte çalışma imkanı yakalamış oluyor .

**Nasıl derseniz,**

Sizin projenizi alıp kullanan ve çeşitli hatalar fark eden kullanıcılar .(Pull Request ) ile yapılan değişiklikleri sizlere gönderiyor ve sizde uygun görürseniz. Yapılan değişiklikler birleştirilip (merge) daha sorunsuz bir koda sahip oluyorsunuz.

Projelerimizi lokalde tutmak yerine uzak bir sunucuda tutmak istiyorsak Github son derece kullanışlı özelliklere sahip. Bu sayede hem bizler uzaktan erişim sağlayarak o proje dosyalarına (Repository) hem de diğer kullanıcılar erişebilecek .

**Önemli bir soru akla gelebilir.**

Ben herkesle paylaşmak istemiyorum ne yapmalıyım derseniz public ve private denilen iki şekilde kaydetmeniz mümkün.

* Public projelere herkes erişebilir.
* Private projelere sadece sizler ve o projenin içinde bulunan diğer katkı sağlayıcılar (contributor)

**Git ile Neler yapılabiliyor ,**

* İstediğimiz bir kullanıcı adı ve mail ile bir repository oluşturmamızı sağlıyor.

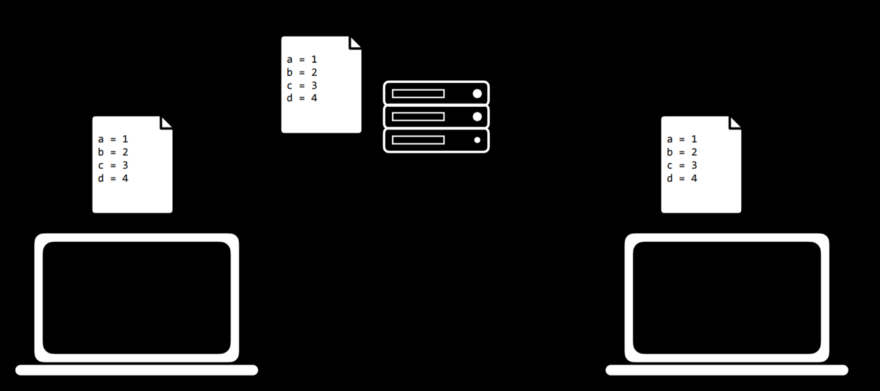
Not: Projelere başlamadan önce , aşağıdaki gibi kim olduğunuz belirten git config komutlarını terminalize ya da git arayüzünü kullanıdığınız başka bir program için de kaydetmeniz gerekiyor .Çünkü yapacağınız değişiklikler de yapan kişi bilgileri olarak bu bilgileri alacak.

git config --global user.name "Talha Kilic"  
git config --global user.email mntalha.kilic@gmail.com

Hali hazırda kayıtlı bir isim var mı diye kontrol etmek isterseniz aşağıdaki komut ile kayıtlı bir isim olup olmadığına bakabilirsiniz.

git config user.name

* Yapılan çalışmaları istediğimiz noktada kaydederek (commit ile ) geriye dönebilmeyi sağlayan noktalar oluşturabiliriz.
* Farklı kişilerin erişimine açma
* Dallar (Branch ) oluşturularak ana yapımıza (master ) zarar vermeden üzerinde uğraşılma imkanı
* Kimlerin hangi tarihte neleri değiştirdiklerini görme
* Senkronizasyonu sağlama

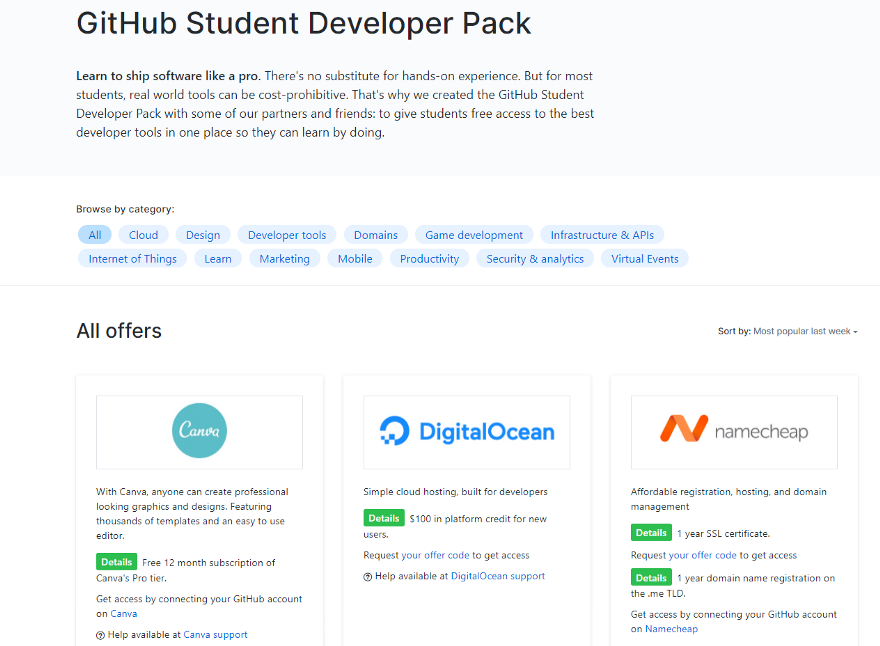
<https://cdn.cs50.net/web/2020/spring/lectures/1/lecture1.pdf>

Senkronizasyon deyince Cs50 slaytlarından alınmış çok güzel bir görüntü var.

Görüldüğü üzere uzak sunucuda var olan a, b,c , d sayıları onu indiren diğer kullanıcılar tarafından da görülebilmekte. Ama önemli nokta senkronizasyon yani sunucuda var olan dosyalar (pull komutu ile) lokale alınıp işleme devam edilebilir .Burda aynı dosya üzerinde çalışmayı sağlar.

Yeri gelmişken Github’un sağlamış olduğu güzel bir fırsattan bahsetmek istiyorum.

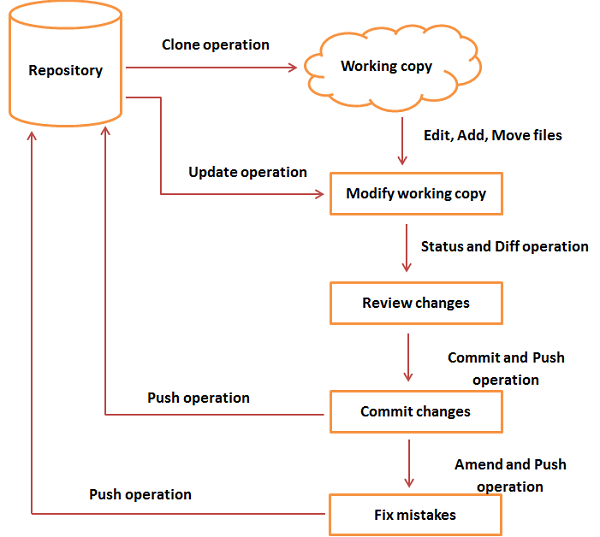
**Student Developer Pack** program sayesinde github bir çok özelliği öğrenci olduğunu kanıtlayacağınız bir öğrenci mailiniz ile github a kaydolursanız ücretsiz olarak sizlere sunuyor.

<https://education.github.com/pack>

Github prodan tutunda üyelik için çeştli ücretler gerektiren birçok site ve uygulamayı ücretsiz olarak kullanma imkanı sağılıyor.Daha detaylı bilgiye resim altında verdiğim linkten ulaşabilirsiniz.

Bir göz atın derim. Şimdi konumuza dönelim.

Git için genel kullanılan life cycle aşağıdaki gibi açıklamak mümkün.

[Git — Life Cycle (tutorialspoint.com)](https://www.tutorialspoint.com/git/git_life_cycle.htm)

Repository sizin ana dosyanız .Projenizi bu dosya içerisinde yerleştiriyorsunuz .Bu dosyayı lokalde saklamak da mümkün Github serverlarında saklamak da.

Sonrasında **clone** işlemi ile o dosyayı lokale çekip üzerinde modifikasyonlar yapmanız mümkün .

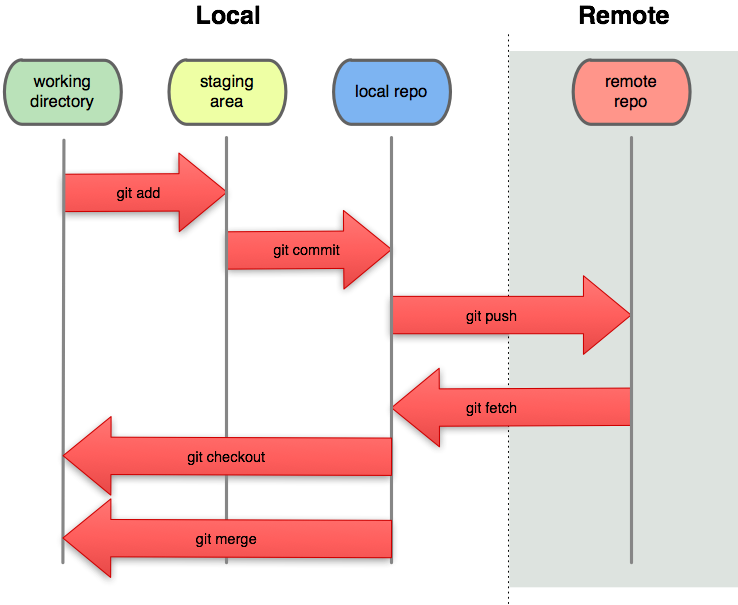
Yapmak istediğiniz düzenleme (edit),ekleme(add) ve taşıma(move) işlemlerini gerçekleştirebilirsiniz.

**Status** komutu ile yapılan değişiklikleri gözden geçirme şansınız var .

Herşey süper ise **add** komutu ile staging areaya (bekleme alanı gibi) yüklenir. Herhangi bir değişiklik yaparsanız tekrardan **add** ile eklemeniz gerekecek.

Sonrasında **commit** komutu ile geriye dönülebilecek bir nokta oluşturuyorsunuz. Commitile **yorumlarınız,** **isim,** **mail**, **tarihiniz** ve o commitin **sha kodu** kaydediliyor. **Log** komutu ile bunları görmeniz mümkün.

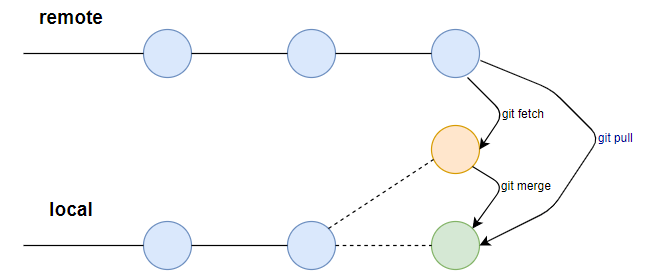
Koyulaştırdığım bu kodları daha temiz bir şekilde yazının sonuna koyacağım.

<https://aht.github.io/whatisgit/#/commits>

Commit işleminden sonra github a yüklemek isterseniz de push işlemi yapılır. Burada yerel alandan uzak olan sunucuya yüklemiş oluruz.

Yukarıdaki resim genel olarak anlatılanların görselleştirilmiş şeklidir.

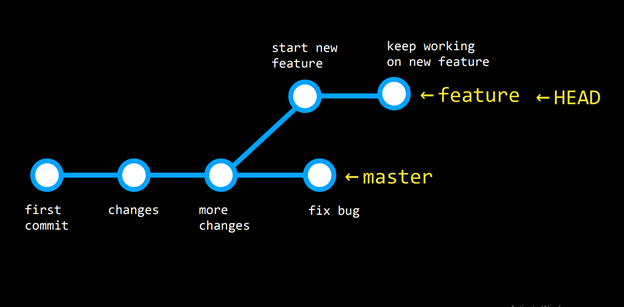
Sonrasında aslında çok önemli bir konu olan **pull** işlemi var şayet remote(Github.com )’da başkaları tarafından bir değişiklik yapıldıysa **pull** işlemi gerçekleştirebiliyorsunuz. Pull işlemi içinde 2 işlem barındırıyor, bunlar **fetch** ve **merge.**

<https://kheri.net/git-fetch-vs-git-pull/>

**Fetch** kısacası uzak sunucudan lokale indiriyor.

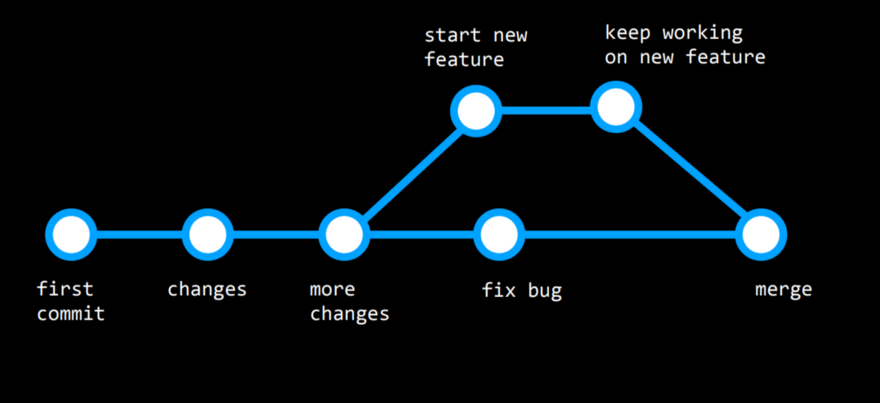
**Merge** ise uzak sunucu da yapılan değişiklikleri ve sizlerin yaptığınız değişiklikleri birleştiriyor. Aynı yerleri her ikisinde de değişti ise conflit durumu oluşur .Buna göre karar vermeniz gerekecek.

Önemli Noktalar ,

<https://cdn.cs50.net/web/2020/spring/lectures/1/lecture1.pdf>

* Ana kol her zaman masterdir ve değişikler en son buraya merge olmalıdr.
* Farklı bir şey geliştirmek için branchler oluşturup burda işlemlerinizi yapıp master ın bozulması tehlikesinin önüne geçebilirsiniz.

Yeri gelmişken Head işleminden bahsetmek istiyorum. Çünkü çok sık karışılacağınız bir durum bu en son neredeyseniz orayı kendine referans alır.

<https://cdn.cs50.net/web/2020/spring/lectures/1/lecture1.pdf>

Sonrasında merge ile de birleştiriyorsunuz.

GİT KOMUTLARI

### git init

Proje ye başlangıç yapmak için yazılan bir kod bu ile birlikte içinde bulunduğun dosyayı git repository olarak kabul eder.

### git add demo.txt

Add komutu ile demo.txt adlı dosyayı staging alanına yükleyebiliyorsun.

### git add file1 file2 file3

Birden fazla dosya eklemek de mümkün.

### git add .

Bulunduğun dizindeki tüm dosyaları eklemek için.

### git commit -m “Initial Commit”

Eklenen dosyaları geri dönülebilir sha kodları oluşturan commit komutu. Bu komut ile yapılan değişikliği yorum “ ” kısmında yazabilirsin.

### git status

Eklenen ve henüz eklenmeyen dosyaları kırmızı ve yeşil renkli gösteren durum raporu kodu.

### git log

O zamana kadar yapılan comitleri ve commit bilgilerinin loglanmış şekilde gösterme.

### git pull

Uzak sunucuda yapılan değişiklikleri çekme.

### git push

Uzak sunucuye yapılan değişiklikleri yükleme.

### git checkout sha\_kodu

Log kısmındaki sha kodu ile önceki komitlere dönebilmeyi sağlar .Tekrardan en son haline dönmek için git checkout master demen yeterli.

Github ile alakalı 1. yazımızın sonuna geldik. Diğer yazılarımızda bu konu ile alakalı daha derin noktalara ineceğiz.

Bu konuyla ilgili herhangi bir sorunuz varsa, linked-in yoluyla bana kolayca ulaşabilirsiniz.