



Wise Quarter
first class IT courses

Git – GitHub



The future at your fingertips

+1 912 888 1630

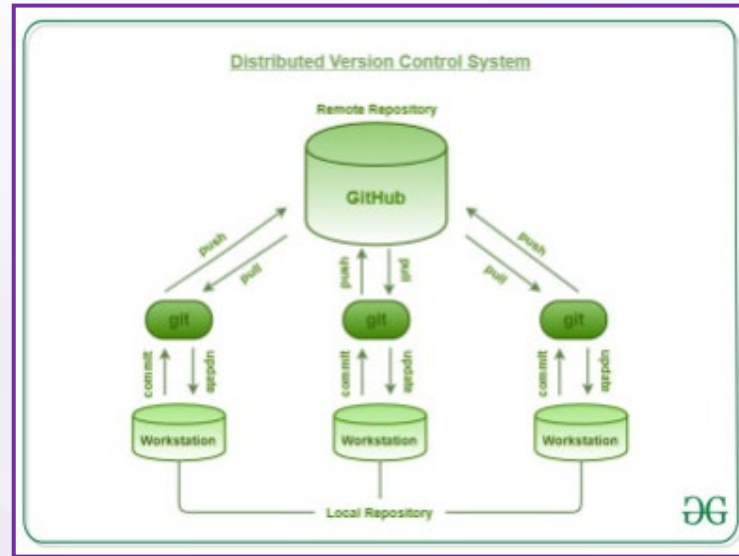
www.wisequarter.com



/wisequarter

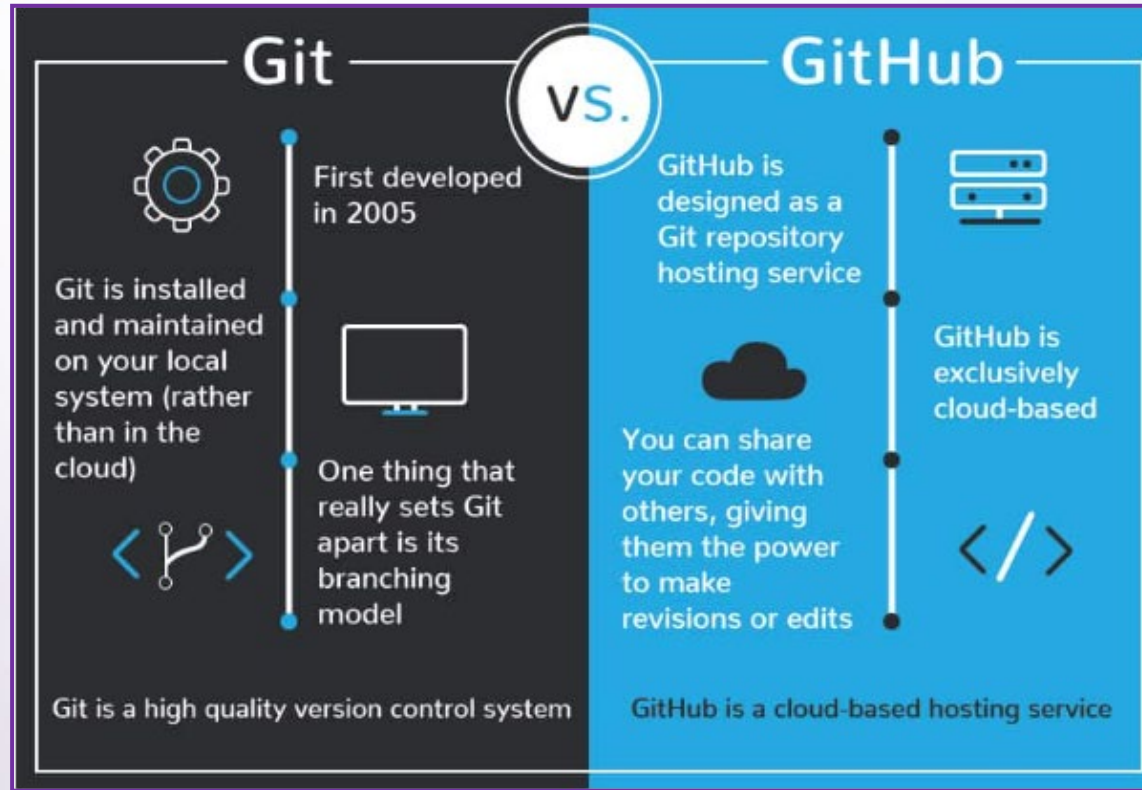
GitHub

GitHub, Git isimli bir version control system(sürüm kontrol sistemi) barındıran cloud tabanlı bir uygulamadır.



Git, bireysel küçük projelerden, ekip olarak çalışılan çok büyük projelere kadar her şeyi hızlı ve verimli bir şekilde yönetmek için tasarlanmış ücretsiz ve açık kaynaklı, **dağıtılmış bir sürüm kontrol sistemidir**(distributed version control system).

Git & GitHub



Git, kaynak kod geçmişinizi yönetmenize ve takip etmenize izin veren bir version kontrol sistemidir.

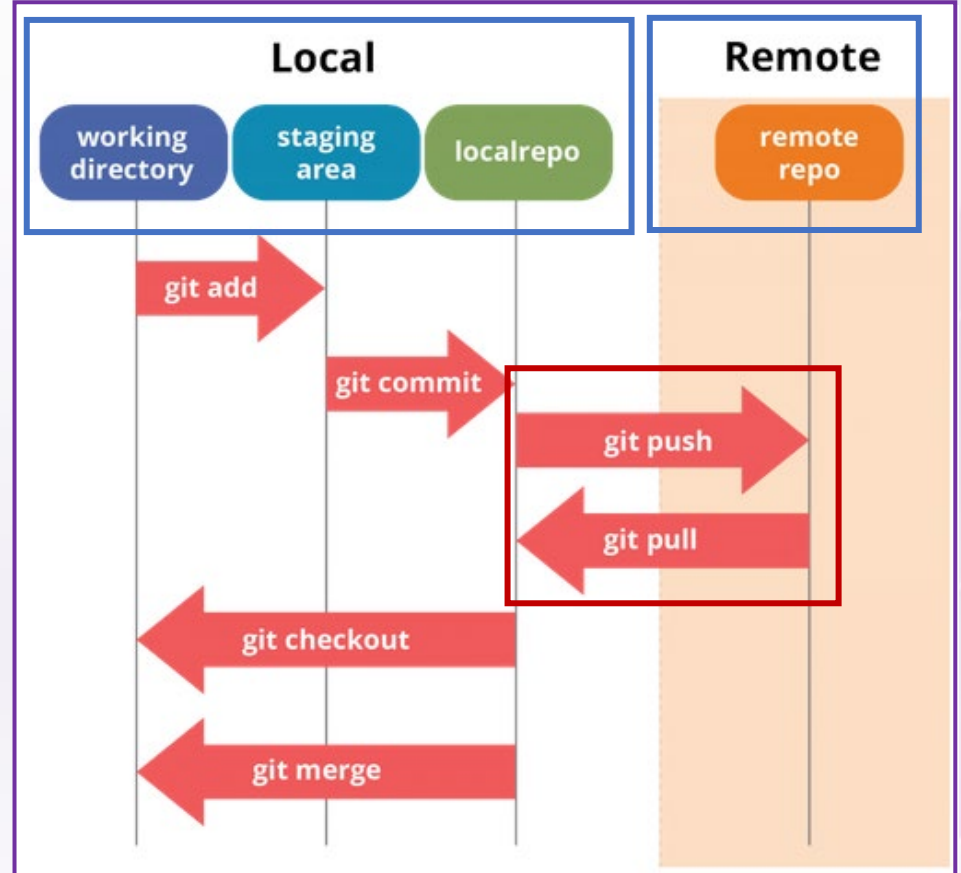
GitHub ise, Git depolarını(**repo**) yönetmenize izin veren bulut tabanlı bir barındırma(**hosting**) hizmetidir.

Tekli Kullanım

Tekli kullanımda her bir projemiz için GitHub'da bir repo oluşturup, bunu istediğimiz aralıklarla bilgisayarımız veya GitHub'daki hali üzerinden güncelleyebiliriz.

GitHub'da yaptığımız değişiklikleri her kaydettiğimizde (commit) Git yeni bir version oluşturur.

Versiyon kontrol sistemlerinin en büyük avantajı, istediğimiz bir anda versiyonlara ulaşabilmektir.



Tekli Kullanım

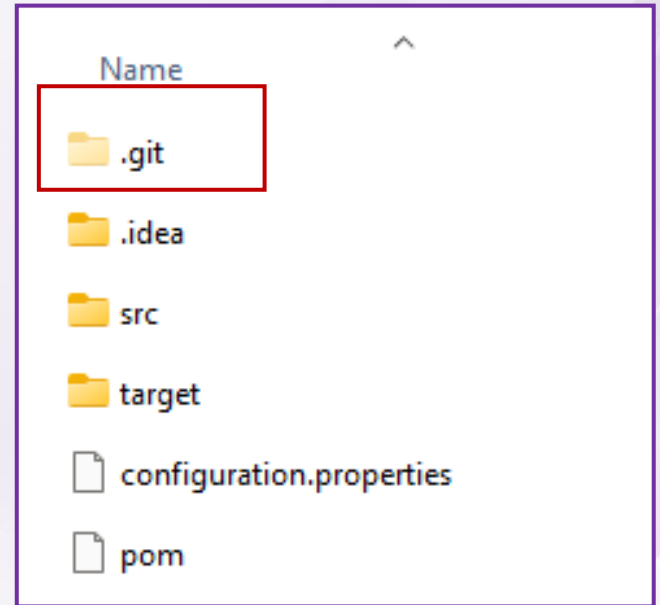
Her değişiklik yapıldığında tüm projenin bir kopyasını alıp kaydetmek çok büyük storage gerektirir.

Git her version'da projenin tamamını değil, bir önceki versiyona göre değişen kodları store eder.

Git version kontrolünü klasör üzerinden yapar, biz IntelliJ gibi bir IDE'de çalışsak da Git o projenin bilgisayarda kaydedildiği klasör üzerinden çalışır.

Git version kontrolünü yapmak için Git'e tanıttığımız (init) her klasöre .git isminde bir klasör oluşturur ve tüm versiyonlar bu dosyada tutulur. Bu dosya silinirse tüm versiyonlar da silinir.

.git dosyasını görmek için bilgisayarınızda “Gizli dosyaları göster” seçeneğinin açık olması gerekir. (Mac için yukarı ok + command + .)



Git Komutlari

Git bilgisayarlarımızdaki windows benzeri uygulamalarla degil, bilgisayarların ilk kullanilmaya basladigi donemdeki gibi, siyah ekrana(command prompt / terminal) yazilan komutlarla calisir.

Gunumuzde GitHub Desktop gibi uygulamalar Command Prompt ihtiyaci olmadan da calismaniza izin verse de version kontrolu icin temel git komutlarini kullanmayi bilmelisiniz.

Essential git commands every developer

should know

- Git Clone
- Git branch
- Git checkout
- Git status
- Git add
- Git commit
- Git push
- Git pull
- Git revert
- Git merge
- Git remote
- Git fetch



Git ve command prompt'daki dosya kullanim komutlarini ezberlemek zorunda degilsiniz. Google ile basit bir arama yaptiginizda tum komutlar ve ne ise yaradiklari karsınıza cikacaktır.

Cmd / Terminal Komutlari

Git bilgisayarlarımızdaki windows benzeri uygulamalarla degil, bilgisayarların ilk kullanilmaya basladigi donemdeki gibi, siyah ekrana(command prompt / terminal) yazilan komutlarla calisir.

	Windows	Mac
Islem	Komut Istemi	Terminal
Klasor hareketleri	cd	
Listeleme	dir	ls
Ekran temileme	cls/clear	
Klasör Oluşturma	mkdir	
Klasör Silme	rmdir	
Dosya oluşturma	echo > dosyaAdi.uzantisi	
Dosyanın içini görme	more	cat
Dosya silme	del	rm
Klasör ve dosya ismi değiştirme	ren	mv

Temel Git Komutlari

git --version : Bilgisayarlarımızda git kurulu ise versiyonunu verir

git config --global color.ui true : Terminalde kullanılan komutlari renklendirir.

git config --global user.name "John Doe": Yaptiginiz islemlerde gorunecek ismi tanimlar.

git config --global user.mail "a@b.com": Yaptiginiz islemlerde gorunecek email'i tanimlar.

NOT : -'den sonra **global** kullanilirsas o kullanicinin tum repolari icin tanimlar
system kullanilirsas, tum kullanicilar ve tum repolar icin tanimlar
local kullanilirsas, sadece o an kullanılan repo icin tanimlar

Local Repo Olusturma

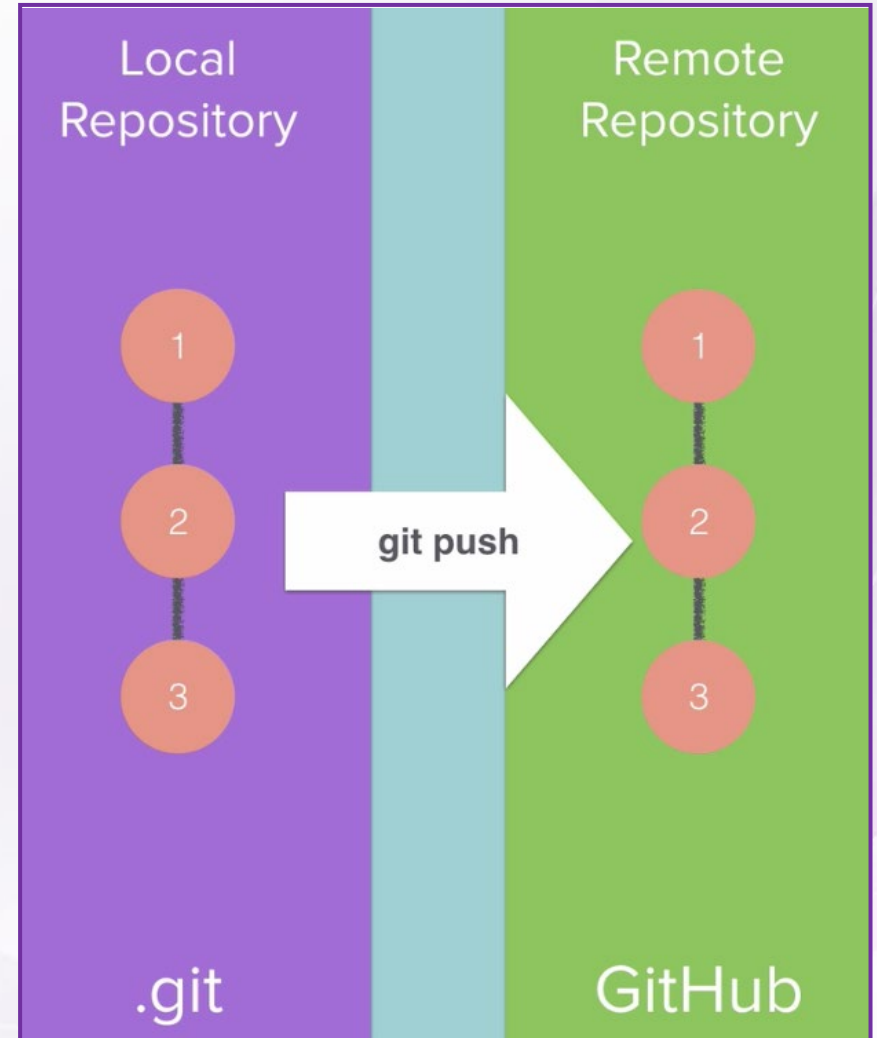
GitHub kullanmasak bile Git ile tüm projelerimizi local'de versiyon kontrol sistemi ile kullanabiliriz.

Bir klasörde git ile versiyon kontrolü yapmak istersek, terminalde o klasöre gidip

git init

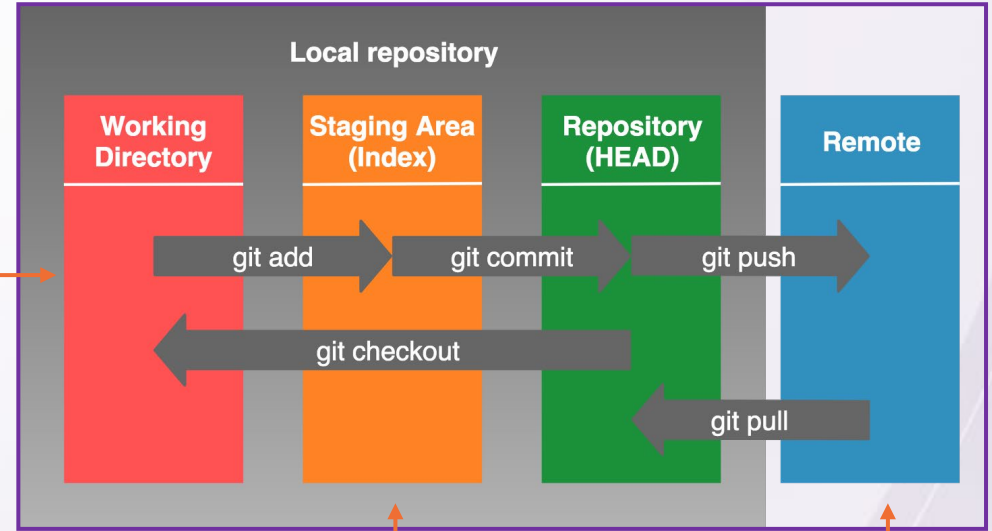
komutu çalıştırılmalıdır.

git init komutu çalıştırılınca git o klasörde hidden **.git** klasörü oluşturur ve bu klasör üzerinden versiyon kontrolü yapar.



Remote Repo'ya Yollama

Working Space (git init)
.git dosyasının bulunduğu klasordur.



Working space'de yapılan değişiklikler git add komutu ile **Staging Area**'ya yuklenir.

Staging Area'daki dosyalar git commit komutu ile gönderilmeye hazır hale getirilir

Remote repo'ya ilk yükleme için, GitHub'da remote repo oluşturulmalı ve local ile remote repo eşleştirilmelidir.

Daha önce GitHub'daki remote repo ile eşleştirilmiş olan projeler için, versiyonu remote repo'ya yollamak istediğimizde **git push** yeterli olur.

Remote Repo'ya Yollama

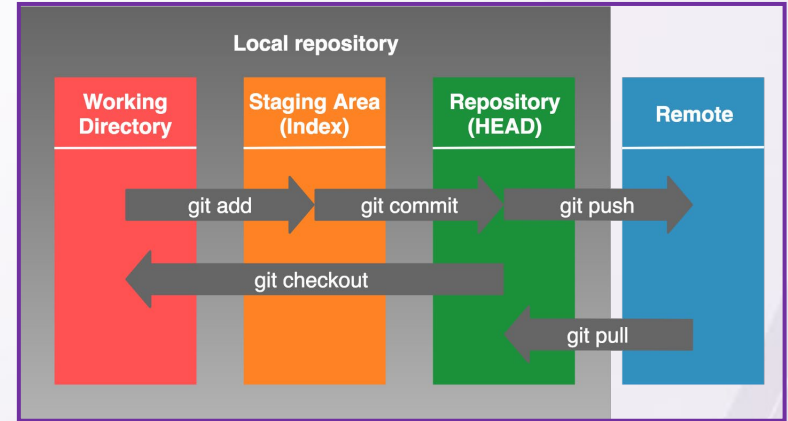
Kisisel kullanimda yapacagimiz tum islemler, kendi bilgisayarimizdaki **local repository** ile GitHub'daki **Remote Repository** arasinda **senkronizasyon**dan ibarettir.

1. Bilgisayarimizdaki bir dosyayi ilk defa Git'e tanitip, Github'da olusturacagimiz bir repo'ya yukleme

- dosyalar git'e ekleyip working space olusturma **git init**
- Dosyalari staging area'ya gonderme **git add** .
- Versiyon olusturup, local repository'e gonderme **git commit -m "commit ismi"**

GitHub'da remote repo olustur.

- Remote repo'daki branch'i tanimla **git branch -M main**
- Remote repo ile local repoyu eslestirme **git remote add origin "remote repo url"**
- local repodaki dosyalari remote repoya yollama **git push -u origin main**





Wise Quarter
first class IT courses

Git – GitHub 2



The future at your fingertips

+1 912 888 1630

www.wisequarter.com

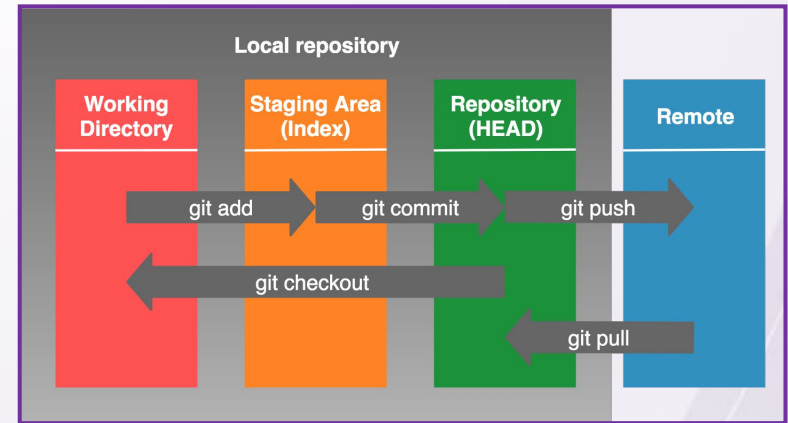


/wisequarter

Remote Repo'dan Kopyalama

2. GitHub'da beğendiğimiz ve erişim yetkimiz olan bir repoyu bilgisayarımıza indirme

- Dosyaları yüklemek için bir klasör oluşturup, git'e ekleme **git init**
- GitHub'da beğendiğimiz reponun url'ini kopyalayip, remote repoyu local repoya indirmek için **git pull remoteRepoUrl**



IntelliJ'de bu işlem için new project menüsünden url ile proje açma seçilir.

NOT : Eger oluşturdugumuz local repo ile remote repoyu bundan sonra da senkronize edeceksek **git pull** yaptıktan sonra local'de commit oluşturup, remote repoya bağlayalım.

- Remote repo'daki branch'i tanımla **git branch -M main**
- Remote repo ile local repoyu eşleştirme **git remote add origin "remote repo url"**
- local repodaki dosyaları remote repoya yollama **git push -u origin main**

Local Repo'da Commit Gecmisi

Git ile yaptigimiz tum version takibi local repoda tutulmaktadır.

Commit yaptigimiz versiyonlar arasinda gezinebilir, istedigimiz commit'e gecebilir ve istersek o versiyon'u projemizin son hali sekline getirebiliriz.

- Local repoda yaptigimiz tum commit'eri listelemek icin **git log**

- Local repoda yaptigimiz tum commit'eri, her commit bir satir olacak sekilde listelemek icin **git log --oneline**

Log listesinde son version HEAD olarak belirtilir.

```
commit 01867b98e9cb888ba78dd16dd52e65abd2cf55d3 (HEAD -> main)
Author: ahmet bulutluoz <abbulutluoz@gmail.com>
Date: Sun Sep 25 07:06:03 2022 +0200

    C03

commit 03c8adde7006b96293a52990e03831af7c8bc73b
Author: ahmet bulutluoz <abbulutluoz@gmail.com>
Date: Sun Sep 25 07:05:22 2022 +0200

    C02

commit 90bd42f51e7692bfc3bd79ca3f06b3aef8d6dd57
Author: ahmet bulutluoz <abbulutluoz@gmail.com>
Date: Sun Sep 25 07:04:41 2022 +0200

    C01
```

```
670da98 (HEAD -> main) D03
3f61cc8 D2
5ad0480 (deneme) D01
01867b9 C03
03c8add C02
90bd42f C01
```

Istenen Commit'e Donme

Git ile projeyi istedigimiz bir commit'e dondurebiliriz.

Eski bir commit'e gitmek icin `git checkout 01867b9`

```
01867b9 (HEAD) C03
03c8add C02
90bd42f C01
```

```
670da98 (HEAD -> main) D03
3f61cc8 D2
5ad0480 (deneme) D01
01867b9 C03
03c8add C02
90bd42f C01
```

Eski commit'e gittikten sonra, o versiyon'u projemizin son hali yapmak icin yeniden versiyon yapmaliz.

`git add .`

`git commit -m "commit ismi"`

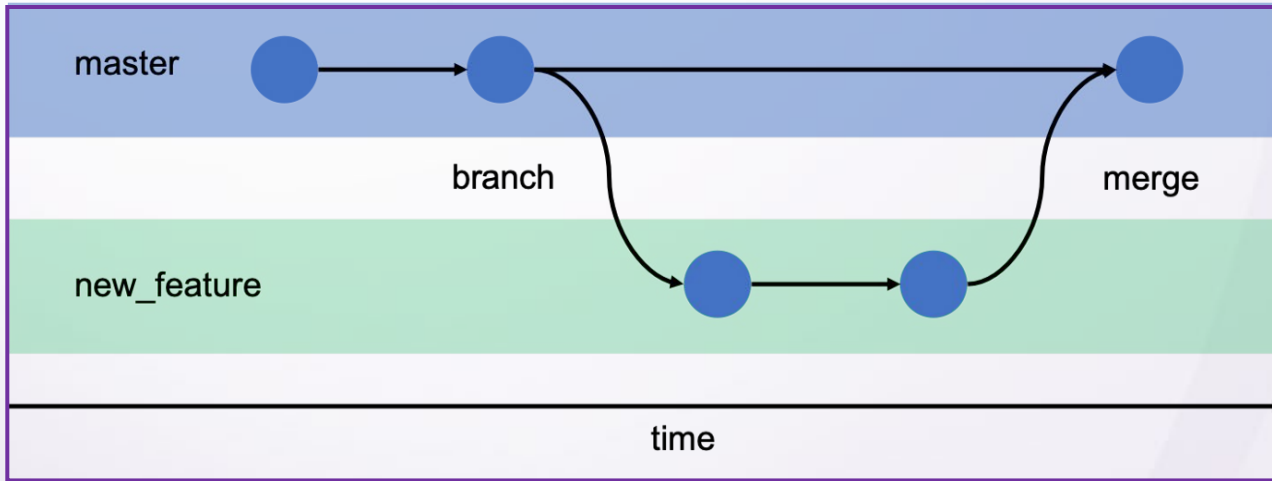
Ancak git bu istegimize uyarı mesajı verecektir. Burada sorumlulugu almamız ve eski versiyona donmek istedigimizi belirtmemiz gerekir. Bunun icin terminale

`git revert 01867b9`

`:wq`

komutlari yazilmalidir.

Local Repo'da Branch Kullanimi



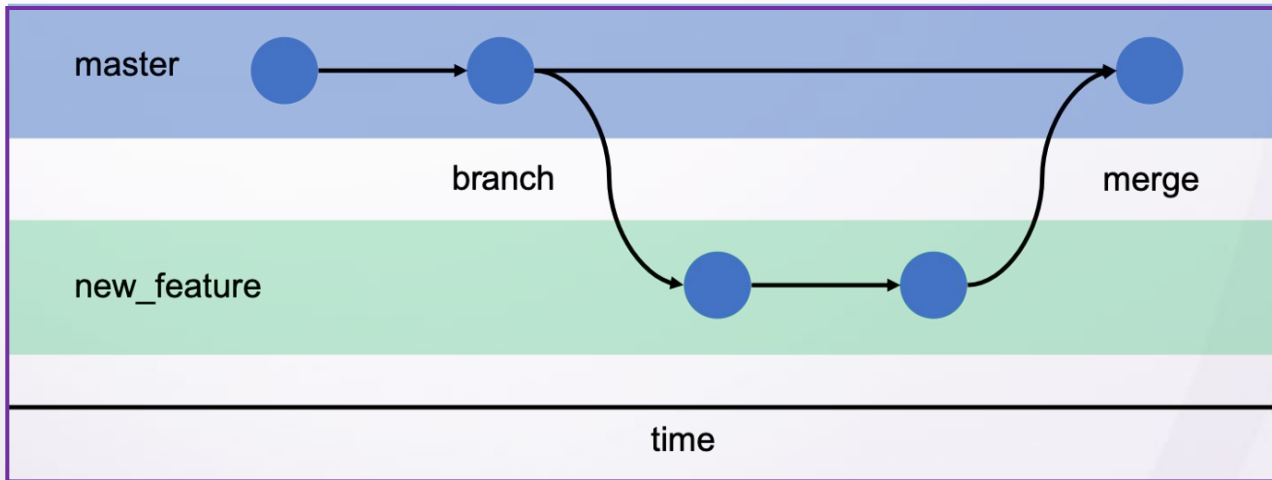
GitHub dagitilmis merkezli bir version kontrol sistemidir.

Bu ozelligi, bir projede calisan pek cok kisinin ayni anda degisiklikler yapabilmesi, kendi kodlarini ana projeye gondermeleri ve yeni kodlar ile ana projeyi guncellemelerine imkan tanir.

Branch (dal) yapisi local repo icin de mumkundur.

Projemizde sonucu belirli olmayan bir sey denemek istiyorsaniz veya projenizde calismaya devam ederken biryandan da bazi degisikliklere calismak istiyorsak main branch disinda bir branch olusturup onunla calisabiliriz.

Local Repo'da Branch Kullanımı



- Branch listesini gormek icin **git branch**
- Yeni bir branch olusturmak icin **git branch branchismi**
- Istenen branch'e gecmek icin **git checkout branchismi**
- Baska bir branch'i kullandigimiz branch ile birlestirme **git merge branchismi**

NOT : Merge yaptigimiz branchlerde ayni kod satirinda farkli kodlar varsa **conflict** olur. Git iki kodu da gosterir ve bizden manuel olarak bunlari duzenlememiz istenir.