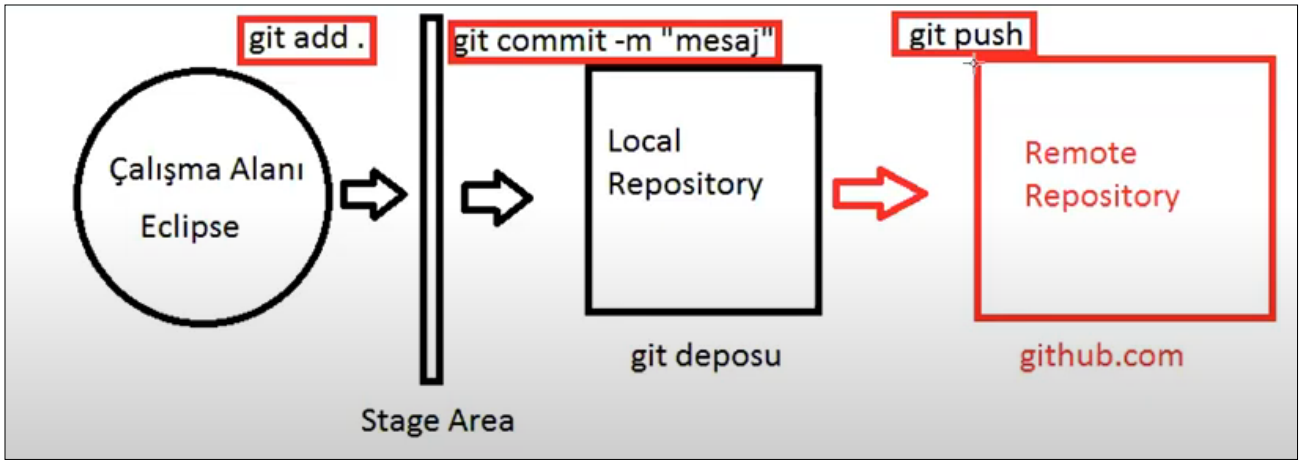
Cmd(Terminal) Commands

```
cd              : klasör değiştirmek için.
cd ..           : bir üst klasöre gidiyoruz.
ls              : klasörün içindeki dosyaları görüyoruz.
dir             : klasörün içindeki dosyaları görüyoruz.(ayrıntılı)
pwd             : bulunduğumuz klasörün adresini görüyoruz.
cls             : cmd ekranını temizliyor.
```

Git Commands

```
git init      : sadece 1 kere, "LOCAL REPOSIROTY" oluşturmak için
git add .     : tüm dosyaları STAGE AREA'ya ekliyoruz..
git commit    : VERSION(commit) oluşturmak için kullanılır.
git push      : dosyaları uzak sucunuya göndermek için kullanılır.

git log       : commit geçmişini gösteriyor
git show      : commit'in ilk 5 hanesini yazarak commit ile ilgili
detayları görebiliyorduk.
```

```
***** Eğer bir CONFLICT oluşacaksa, siz yeni bir sürüm çıkarmış
olacaksınız, uzak sunucuda da yeni bir sürüm çıkmış olacak. YANİ
COMMIT SAYINIZ EŞİT OLACAK. İşte conflict o zaman ortaya çıkar. Çünkü
conflict dediğimiz şey AYNI VERSİYON ÜZERİNDE, AYNI SATIRA FARKLI
KODLAR YAZILMASIDIR.


***** Eğer github.com'da, localimizden daha fazla ve daha yeni bir
versiyon(commit) varsa, bilgisayarımızdaki kodları github.com'a
gönderemeyiz !!!!! Önce son halini, güncel halini remote'tan(github)
almak zorundayız. Daha sonra gönderebiliriz..

-----------------------------DERS NOTLARI-------------------------

***** Her değişiklikten sonra push etmek gerekiyor mu ?

Neden push edilir ? : Eğer dosyaları uzak sunucuya(github vb)
göndermek istiyorsak, push ederiz.

git fetch    : github.com'daki verileri localimize alıyoruz.

git merge    : eğer herhangi bir CONFLICT(çakışma) varsa, onu
çözmemize yardımcı olur.

git pull     : github.com'dan aldığımız projedeki değişiklikleri
localimize uyguluyor.
```




```
Soru : Aynı proje üzerinde birden fazla kişi çalışıyorsa, ortaya
çıkacak karışıklıklar nasıl çözülebilir ?
    - herkesin farklı bir görevi olur.
    - sürekli iletişim halinde kalınabilir.
    - önce herkes kendi işini tamamlar ve daha sonra havuzda toplanır.
    - sürekli bir hareket olmalı, proje üzerine sürekli konuşulmalı..
```

Branch (Feature)

```
branch(dal) : herkes kendi çalışma alanında çalışsın !
```

```
Branch'in ne gibi faydaları olabilir ?
 * Karışıklığı önler.
 * Hata riskini azaltır.
 * Risksiz çalışma imkanı oluşturur.
 * Geri dönüş şansı verebilir.
 * Kodları test edebilme imkanı tanır.
 ***** İşimiz kolaylaşır.
 * Alternatifleri deneme imkanı verir.
 * Organize olmayı sağlar.
 * Ana projeyi koruma imkanı sağlar.
 * Ortak çalışmayı daha verimli hale getirir.
```

git branch : halihazırdaki tüm branchleri ve şuan üzerinde bulunduğumuz branchi görebiliyoruz.
git branch dalIsmi : bu şekilde yeni bir dal oluşturabiliyoruz.
git checkout dalIsmi : bu şekilde istediğimiz dala geçebiliyoruz.
git push --set-upstream origin dalIsmi : bu şekilde -sadece 1 kere- oluşturduğumuz dalı github.com'a gönderiyoruz.
git merge dalIsmi : dalIsmi branchini, üzerinde bulunduğumuz dal ile birleştiriyorduk.

*** herhangi bir dal üzerinde yeni bir dal oluşturursak, üzerinde bulunduğumuz dalın o kısma kadarki kodlarını kullanabiliyoruz.

Şirketlerde ortak çalışma nasıl oluyor ?

* Localde bir MASTER ve FEATURE branchleri oluyor..
* Github'da da MASTER ve FEATURE branchleri oluyor..
* Biz hiçbir zaman MASTER branche kod eklemeyiz. Biz kodlarımızı FEATURE branche yazarız ve onu github.com'a göndeririz.

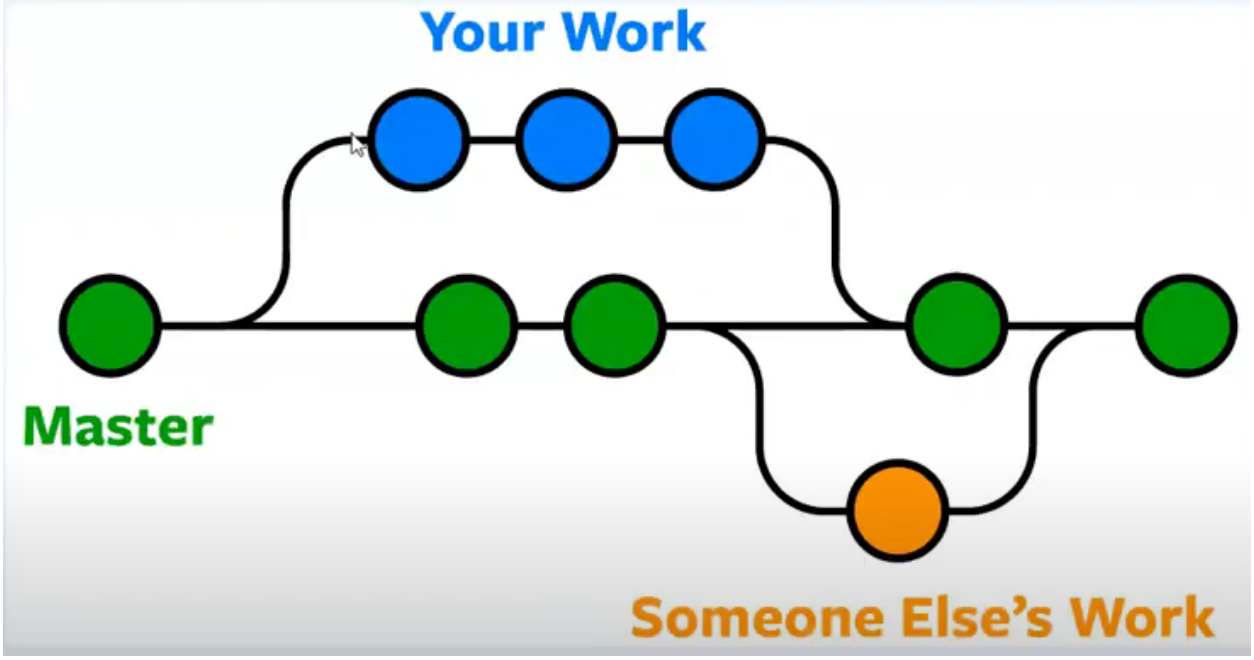
* Team Lead, github.com'da karşılaştırma işlemi yapar. Eğer bizim kodlarımızı uygun görürse MERGE yapar ve MASTER branch ile birleştirir. Eğer uygun görmezse, bizden değişiklik yapmamızı ister, biz de değişiklik yapar ve daha sonra tekrar göndeririz..

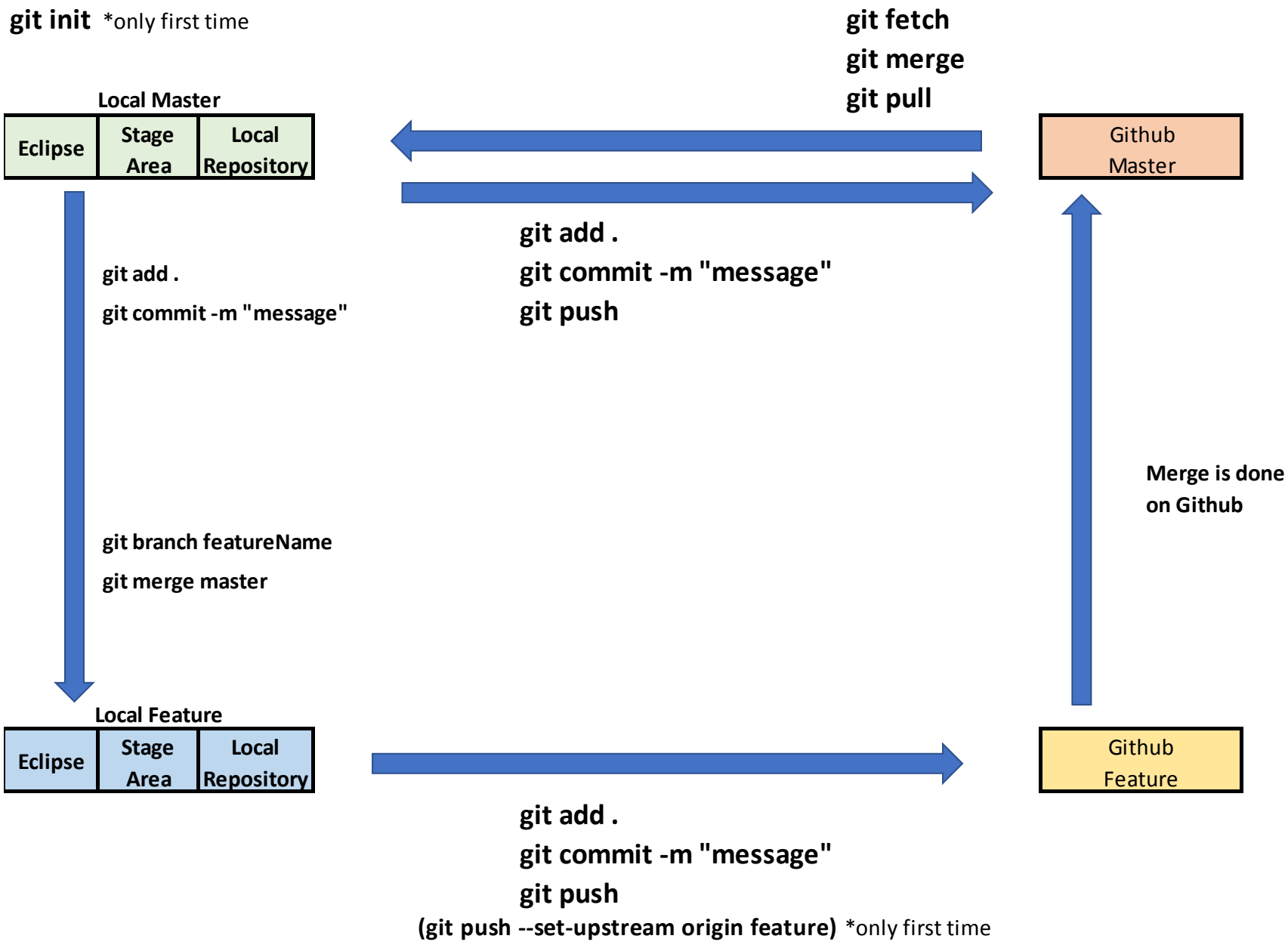
* MASTER güncellendikten sonra, Team Lead herkese MASTER branchlerini GÜNCELLEMELERİ GEREKTİĞİNİ söyler, herkes günceller ve kodlarını yazmaya devam eder..

git status : herhangi bir yerde çalıştırabilirz. Yaptığımız değişikliklerin, şuanda yapılması gerekenlerin vs. durumunu bize gösteriyor.

git reset --hard origin/master : localinizdeki herşeyi siler. ve github.com'daki güncel hali direk getirir projenize yapıştırır. bir nevi gözü karartma hali..

.gitignore : bu dosyanın içerisine, github.com'a göndermek İSTEMEDİĞİMİZ dosyaların ve klasörlerin isimlerini yazıyoruz.


Your Work
Master
Someone Else's Work


git init *only first time
git fetch
git merge
git pull
Local Master
Eclipse
Stage Area
Local Repository
Github Master
git add .
git commit -m "message"
git push
git add .
git commit -m "message"
Merge is done on Github
git branch featureName
git merge master
Local Feature
Eclipse
Stage Area
Local Repository
Github Feature
git add .
git commit -m "message"
git push
(git push --set-upstream origin feature) *only first time