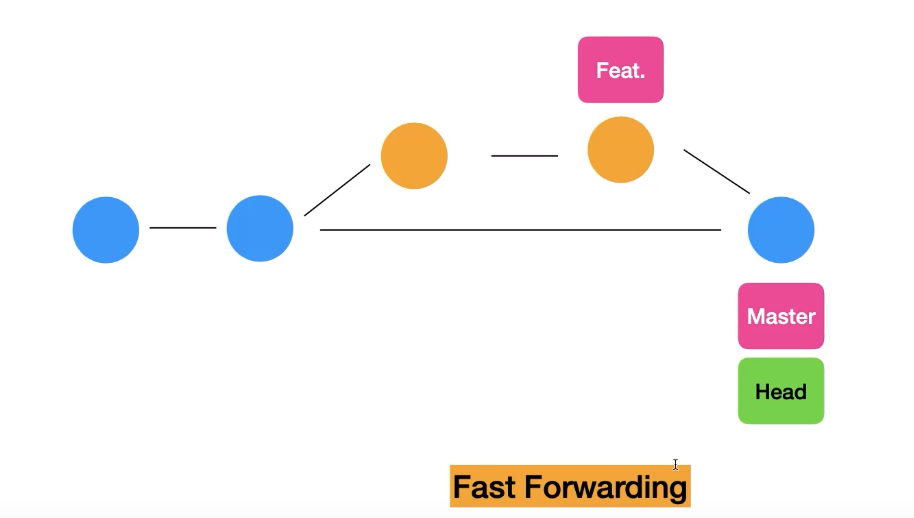
***Git Not Defteri***

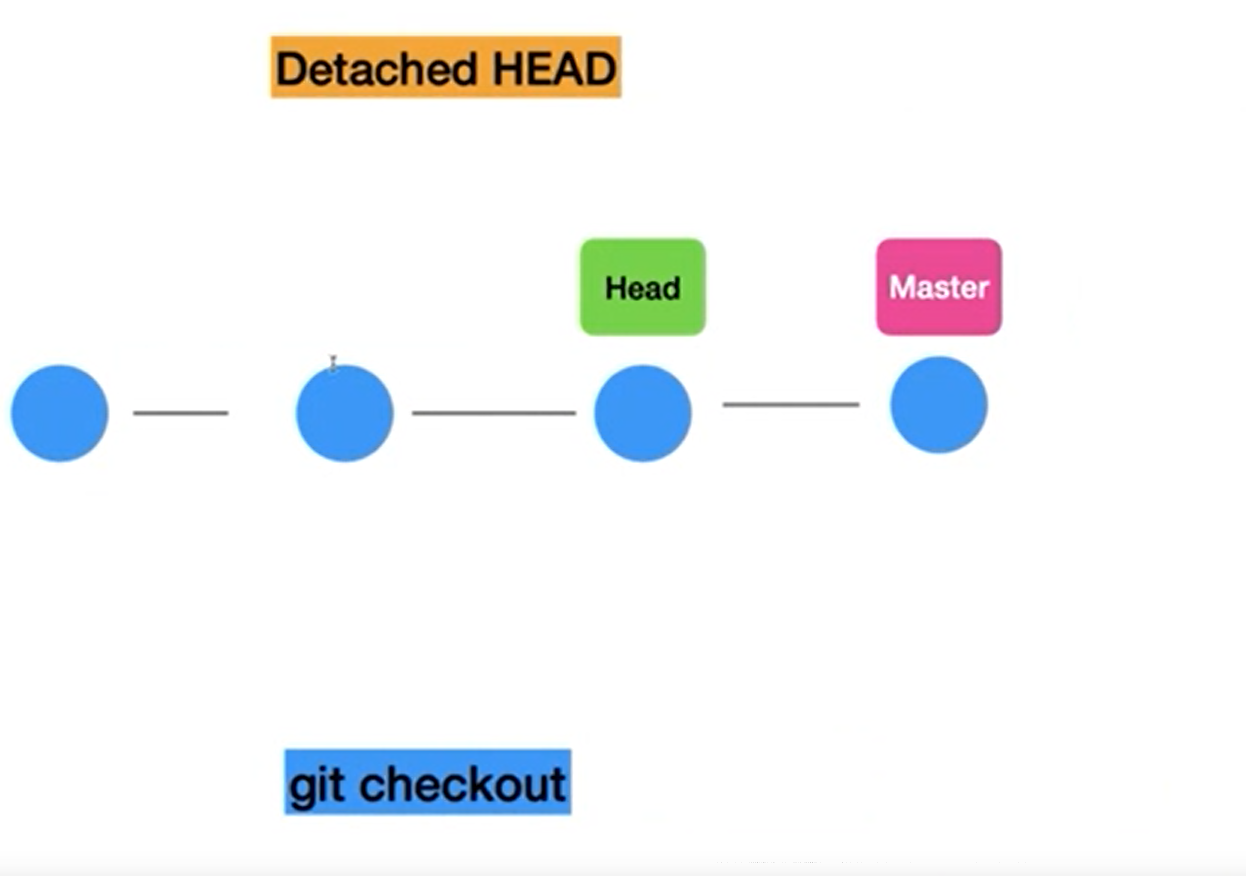
***-Terminal Kullanımı***

* *ls -> “list” bulundugunuz directoryideki klasörleri gösterir(ls –lf gizli dosyalarıda gösterir).*
* *pwd -> “present working directory” bulunduğunuz klasörü gösterir.*
* *cd <filename> -> “change directory” klasörler arası geçiş yapabilmeyi sağlar ‘cd ..’ kullanımı bir geri boyuta geçmeyi sağlar.*
* *mkdir <filename> -> “make directory” klasör oluşturmayı sağlar.*
* *touch <filename> -> dosya oluşturur.*
* *rm <filename> -> “remove” dosya siler.*
* *rm –rf <filename> -> seçilen klasörü onay almadan siler (-Ri onay ister) .*
* *git config –list -> git config ayarlarınızın bulunduğu listeyi verir.*

***-Git Temelleri***

* *Git init -> initialize(başlatmak) klasöre git i kurar (gizli dosya halinde ekler giti silmek isterseniz git klasörünü silebilirsiniz)*
* *git status -> dosyların,klasörlerin ne durumda olduğunu gösterir(hangi branch,izleniyor mu izlenmiyor mu,staging areda mı vs.)*
* *git commit –m “<message>” -> staged(index) aşamsında olan dosyaları commit eder.*
* *git log -> ettiğiniz commitlerin detaylarını gösterir.*
* *.gitignore dosyası -> bu dosya commit etmek istemediğiniz dosyaları belirtmenize ve onları görmezden gelmenize yarar dosyanın içine yazılan dosyalar veya klasörler stage’e gönderilmez*
* *master ve HEAD kavramları -> master kavramı main branchı belirtir head ise bulunduğumuz commiti ifade eder*
* *git branch -> açılmış branchleri gösterir.*
* *git branch <branch\_name> -> branch oluşturur.*
* *git switch <branch\_name> -> belirtilen branche geçer.*
* *git merge <branch\_name> -> “birleşmek” branchleri birleştirir ve yeni bir commit olarak ekler.*
* *NOT: fast-forwarding -> bu bir terim master branchından ilerlemek yerine başka bir brancden ilerleyip onu masterda birleştirmeyi ifade eder*
* 
* *Conflickt -> merge yaptığın branchlerin çakışması olayına denir.*
* *git restore <file\_name> -> seçilen dosyayı en son commitdeki haline çevirir.*
* *git stash -> “depo” bulunulan branchde commit edilmemiş değişiklikleri ayrı biryere alır bu sayede farklı bir branche geçmen gerektiğinde commit almadan stash’e atıp geçebilirsin ardından geldiğinde git stash pop komutu ile geri çağırabilirsin(sadece son commitden bu ana kadar yapılan yani commit edilmemiş değişiklikleri stash’e atar)*
* *git stash pop -> bulunulan branchdeki stash edilen değişiklikleri geri çağırır.*
* *git stash list -> stash deki değerleri gösterir (stash@{0}, stash@{1}..) gibi her stashde bir index artar bir stashde birden fazla dosyadaki dğeişikliği stashe atabilirsin.*
* *git stash apply <stash\_id> -> verilen stash id’yi(tash@{0}, stash@{1}..) geri çağırır değişiklikler dosyaya geri gelir ama stash listden stash popdaki gibi silinmez.*
* *git stash apply -> burda ise bütün stashleri geri yükler ama liteden hiçbirini silmez.*
* *git stash clear -> stash listesini temizler.*

***-Geçmişe Dönme***

* *git checkout <commit\_hash>* ***->*** *girilen commit hashine dönmeyi sağlar ama döndüğünde önünde kalan commitler silinmez ve HEAD bağımsız olur buna detached HEAD denir allta örnek verildi.*
* 
* git reset <commit\_hash> -> girilen commitden sonrasını siler ancak yapıln değişiklikler kalır.
* git reset ––hard <commit\_hash> -> girilen commitden sonraki commitleri değişikleriyle birlikte siler.
* git revert <commit\_hash> -> bulunduğumuz commitedn girilen commitde yapılan değişiklikleri siler ve revert edilen comiti silip yeni commit oluşturur.
* git diff -> en son commit ile stage aşamasına atılmamış değişikliklerin farkını gösterir ,stage aşamasına atılırsa göstermez!(git add).
* git diff <commit\_hash> -> girilen commit ile şuanda bulunulan commitin arasındaki farkları gösterir.
* git diff <commit\_hash> <commit\_hash2> -> birinci hasden ikinci hash arasındaki farkları gösterir
* Not : bulunduğunuz commiti ifade etmek için HEAD tagını kullanabilirsiniz örn: git diff HEAD bu durumda değişikliklerinizi stage’e atmış olsanız bile en son commitle arasındaki farkı gösterir.
* git rebase <branch\_name> -> girilen branchi bulunan branche rebase eder yani girilen branchdeki commitleri tek tek ekler buda mantıksal olarak daha anlaşılır olur mergedeki gibi bütün commitleri tek bir commite sığdırırsak ilerki zamanlarda anlamada güçlük çekebiliriz :)