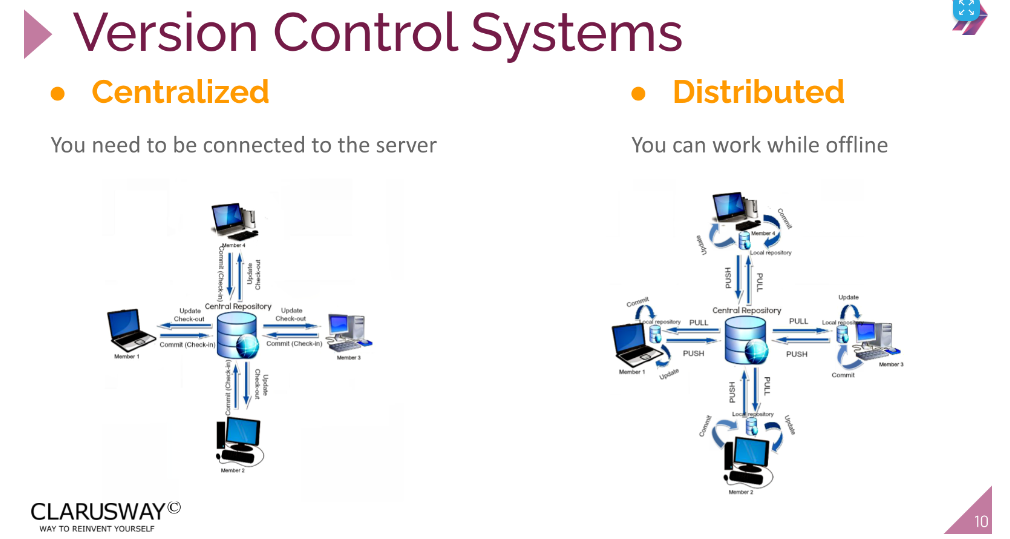


Version kontrol sistemi versiyonlardaki değişiklikleri kontrol eder. Biz Git kullanacağız. Git bu değşikliği fark eder, kaydetmek istiyor musun diye uyarır.

Bir dosya veya klasörün sınırsız şekilde versiyonlarına ulaşabiliyoruz. İstediğin zamana dönebiliriz.

Bir dosyaya, bir klasöre farklı farklı kişiler müdahale edebiliyor. Birden fazla kişinin aynı dosya üzerinde çalışmasını sağlar.

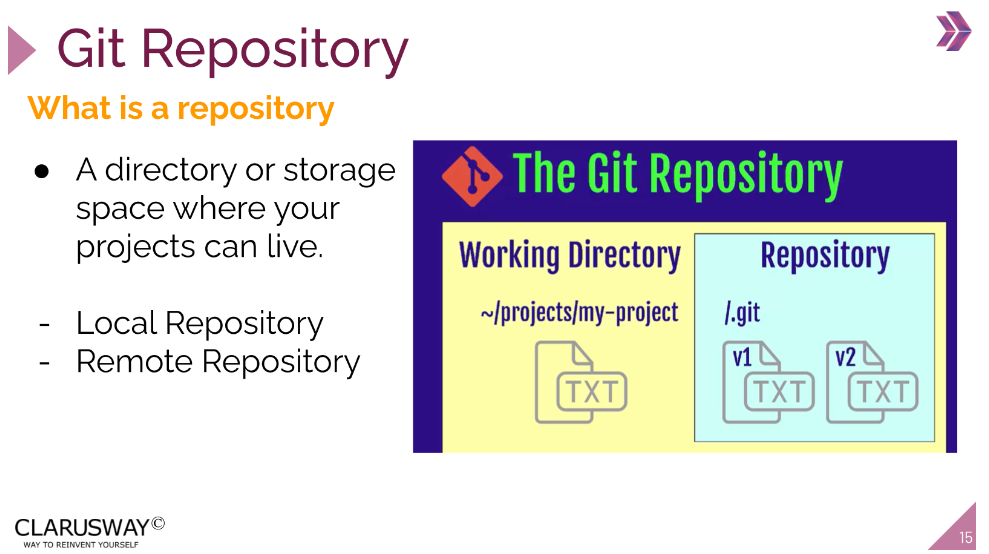
Versiyonu almak bizim verdiğimiz bir komutlar oluyor. Versiyonda ne değişikliği yapıldığını, ne zaman yapıldığını, neden yapıldığını ve kimin yaptığını görebiliyoruz.



Centralized Server sisteminde sistemde herkes servera bağlanıyor ve bu server üzerinde çalışıyor. Proje üzerinde çalışmak isteyen herkes internete bağlı olmak zorunda.

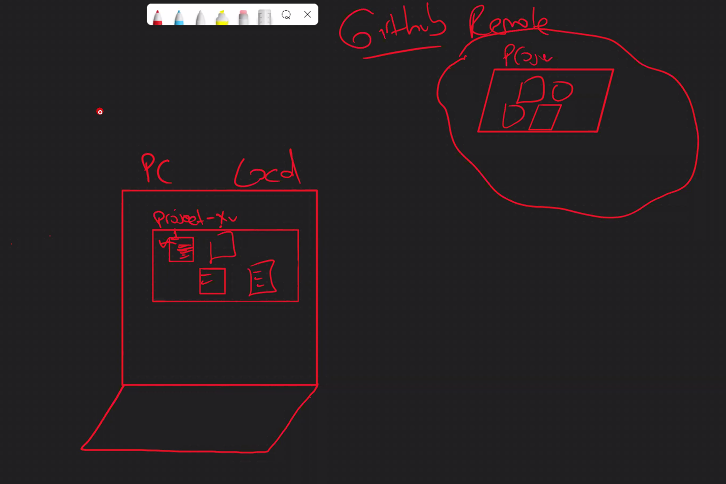
Distributed Server sisteminde serverdan projeyi bilgisayara çekip offline olarak çalışıp sonra internet olduğunda sisteme yüklüyoruz. Öte yandan ana server çökse bile başka bilgisayarlarda indirilmiş en son versiyon elde olacaktır.





Repository nedir? İki git reposu var.

Biri local diğeri remote:



Bash, linuxun terminali

Git Bash Git le gelen bir terminal tool. Bu bir emilatör, windowsta sanki linux makineymiş gibi komutlarımızın çalıştırılmasını sağlıyor.

Git Bash te daha önce yazdığımız komutları görmek için üst tuşuna basacaz. Basınca komutlar sırasıyla geliyor.

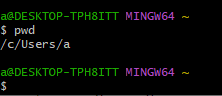
Boşluk yerine alt çizgi kullanmayı alışkanlık haline getirmeliyiz.

Boş bir klasöre regular klasör diyoruz.

Dosyaları görmek için kullanacapımız komut ls dir.

Hangi dosyanın içinde olduğumuzu gösteren komut da pwd dir.

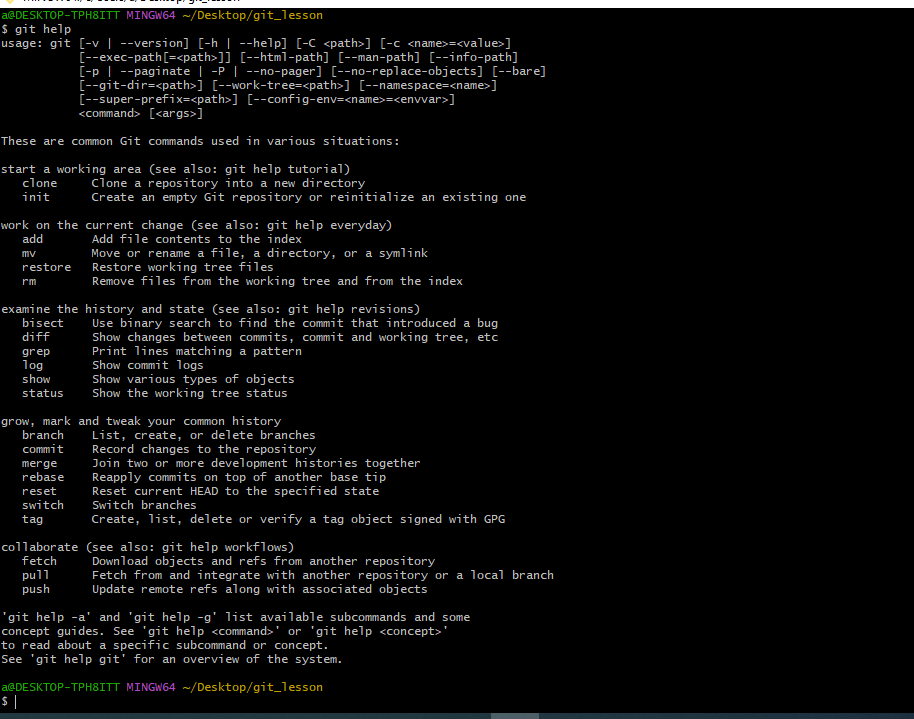
Klasörü değiştirmek için cd kullanıyoruz. (change directory, directory klasör)



Buradaki sonsuzluk işareti kullanıcı kısmında olduğumuzu simgeler.

Boş zamanlarınızda temel linux komutlarını çalışın.

“git help” komutuyla git le hangi komutları kullanabilirim onu gösterir:

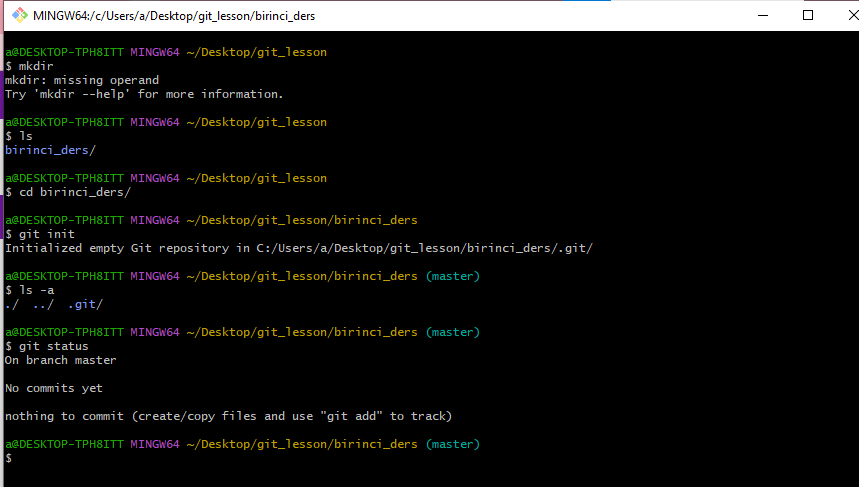


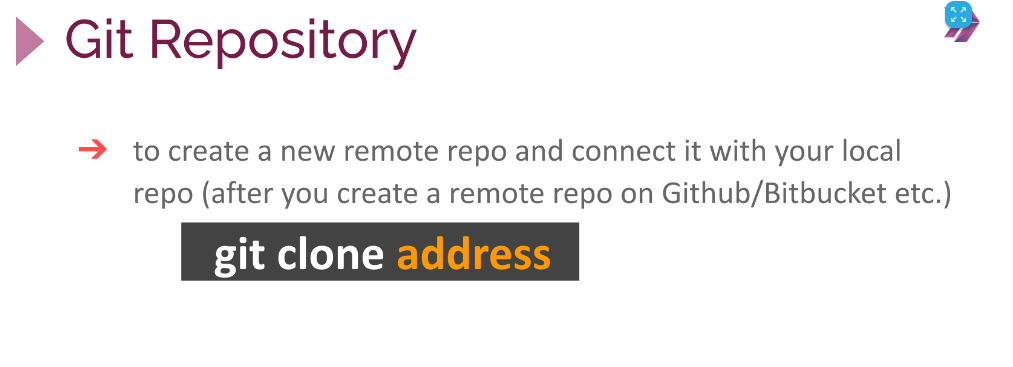
Linux komutlarında “mkdir” yeni klasör oluşturur.

Linux sisteminde noktayla başlayan dosyalar gizli dosyalardır. Ls komutu gizli dosyaları göstermez. Bunun için ls –a komutunu kullanmamız gerekir.

"git init" komutunu girdiğimizde .git şeklinde bir gizli klasör oluşturuldu. Bu bir local repo dur. Bu şekilde bir local repo oluşturduk. Bu birinci yöntemidir.

"git satus" komutu



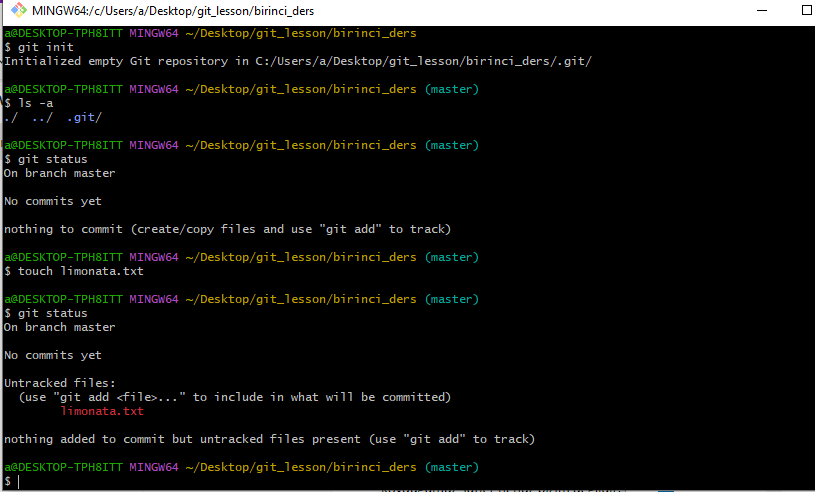


Git de repo oluşturmanın ikinci yöntemi ise klonlamaktır. Klonlamakta daha yaygın olan bir yöntemdir. Github da oluşturulan remote repo bilgisayara kopyalanır.



Bilgisayara baktığım gördüğüm alan yani içinde olduğumuz klasör working directory dir. Amacımız working directory de yaptığımız çalışmaları depolamaktır.

Yaptığımız dosyaları direk repoya koyamıyoruz. Şöyle ki süpermarkete gittiğimizde alışveriş sepetine koyduğumuz sütler henüz bizim değildir. Ücertini verene kadar bizim değildir. Ancak başkası da gelip almaz. Staging area işte o market sepetidir.



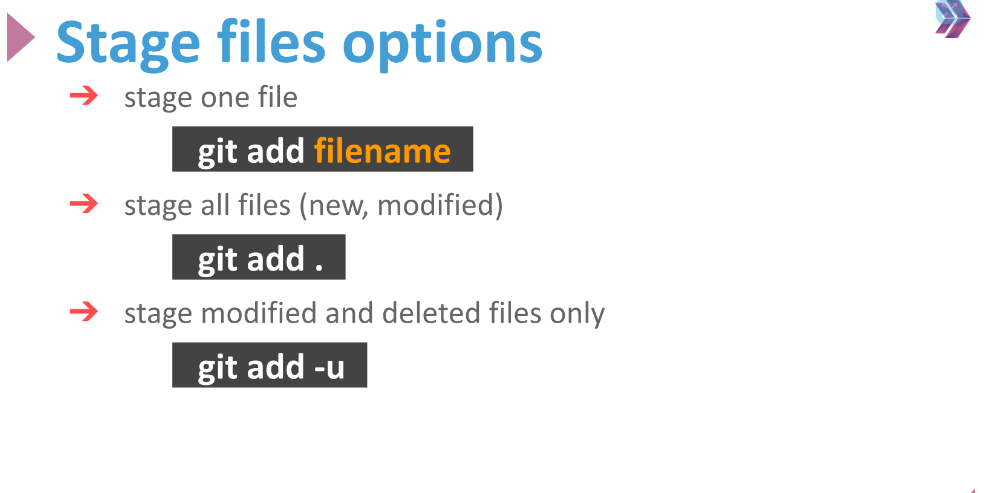
touch limonata.txt komutuyla yeni bir dosya oluşturduk.

Önceki git status komutunda bir şey görünmemişken dosya oluşturduktan sonra track edilmeyen bir doysa olduğunu belirtti.

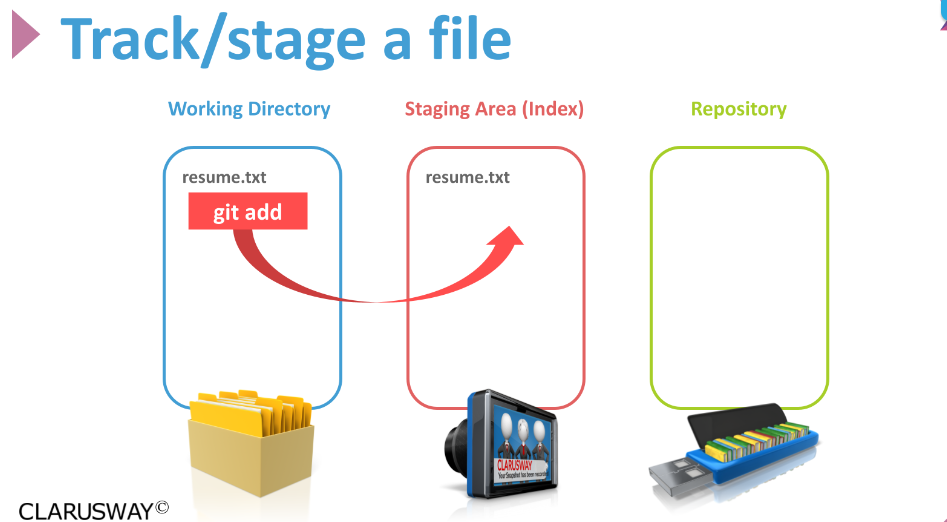
Takip edilmeyen demek repoda versiyonlanmayan bir değişiklik yapılmış. Track etmek için git add komutunu kullan diyor.

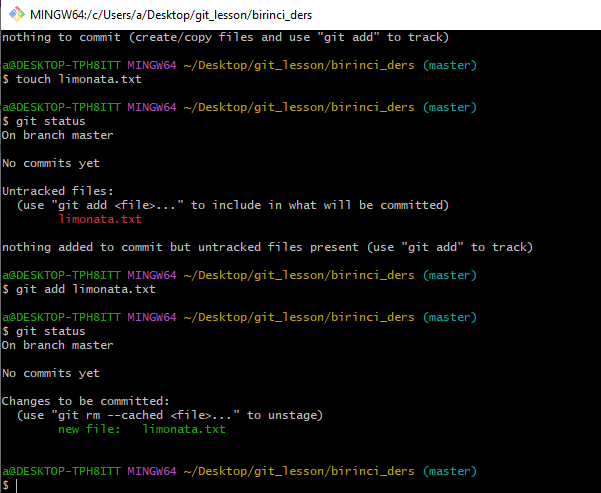


Biz bu dosyayı direk repoya atamayız. Önce staging area ya atmamız gerekiyor.



"git add limonata.txt komutuyla bu dosyayı staging area ya attı:

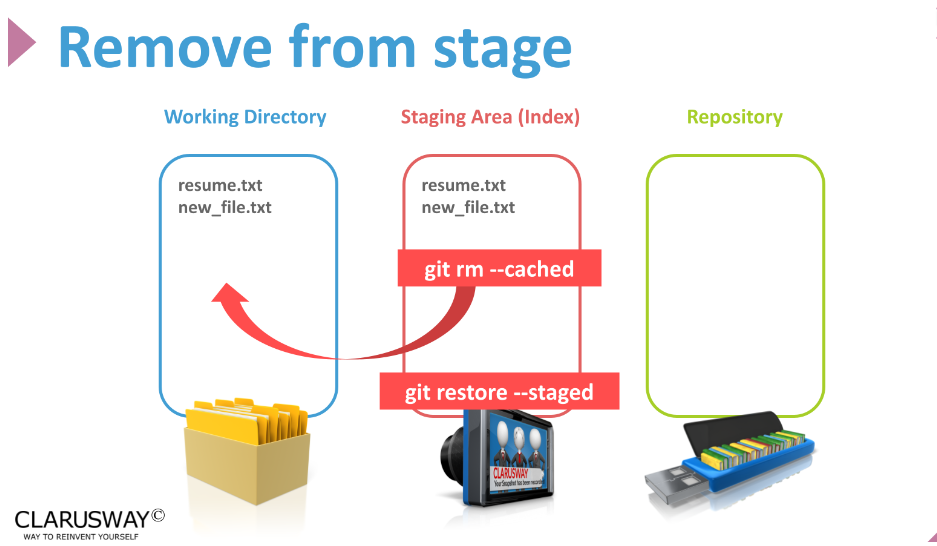


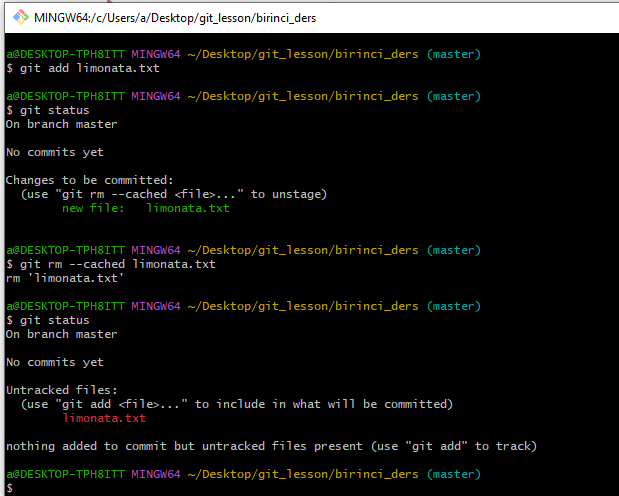


bu kez "git status" komutunu girdiğimizde Staging area ya giren dosya yeşil renkle gösterilir.

Bize diyor ki commit etmek istemiyorsan bu dosyayı "git rm - -cached" komutuyla unstage edebilirsin.

Burada rm remove demek. Bu komutu girdiğimizde staging areadan çıkarır:

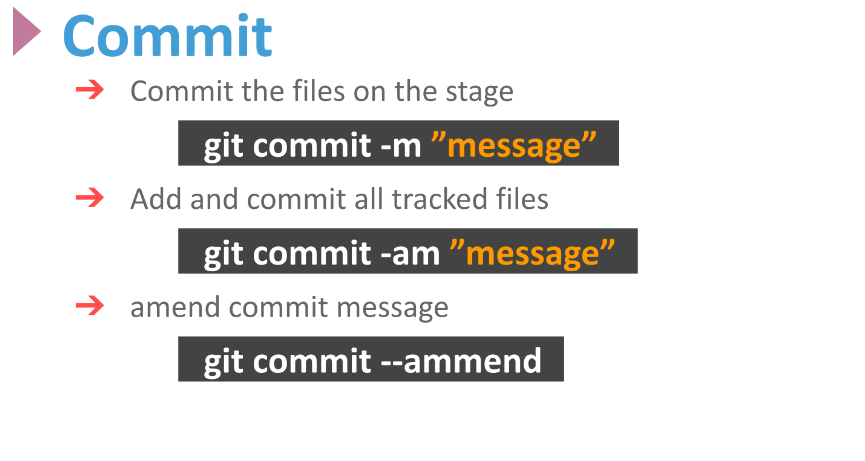


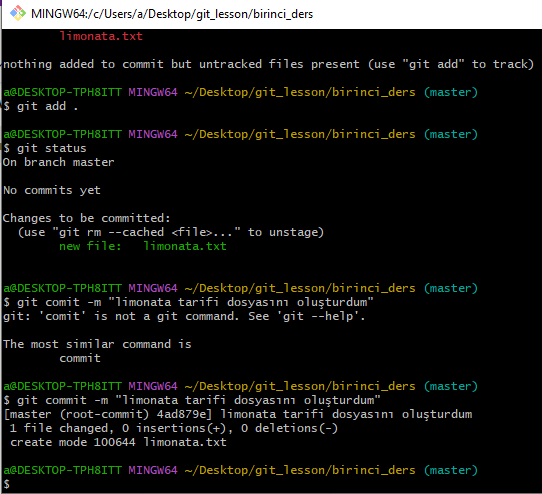


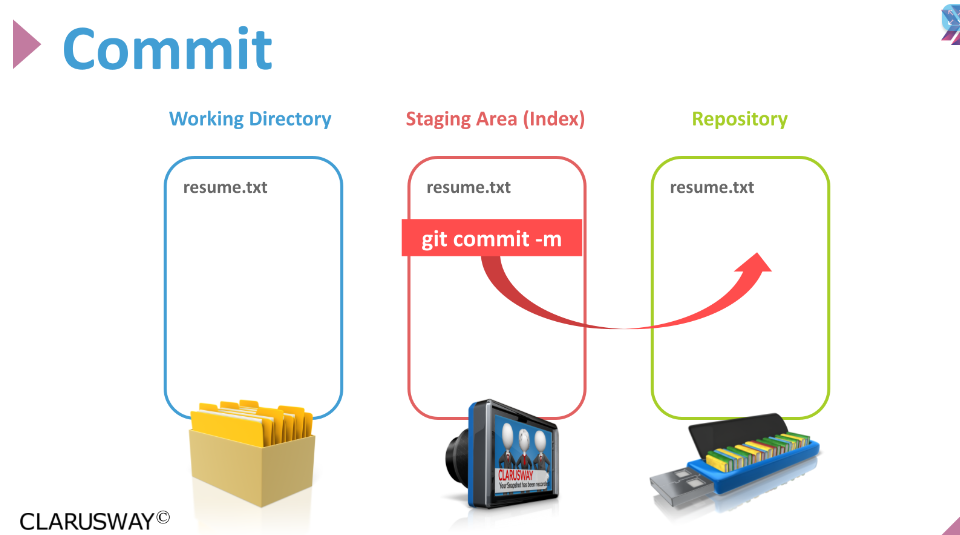
Burada ilk önce staging areadan çıkardık ve "git status" komutuyla kırmızı renkli olduğu için staging areadan çıkardığını gördük.

Bu kez git add . Komutuyla (bu komut bütün dosyaları staging area ya ekler) tekrar staging area ya ekledik.

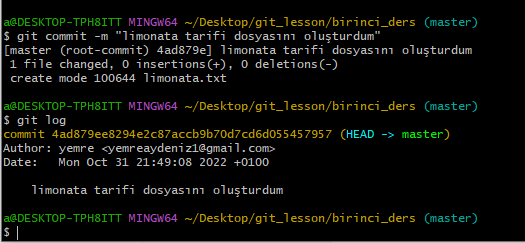
Şimdi limonata dosyasını git commit -m "" komutuyla repo ya ekleyeceğiz.  Tırnak içerisine neden commit ettğimizi açıklayacak bir ifade yazıyoruz:



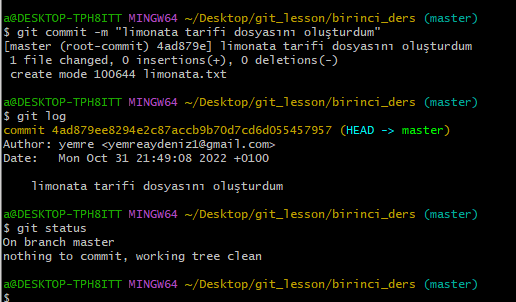




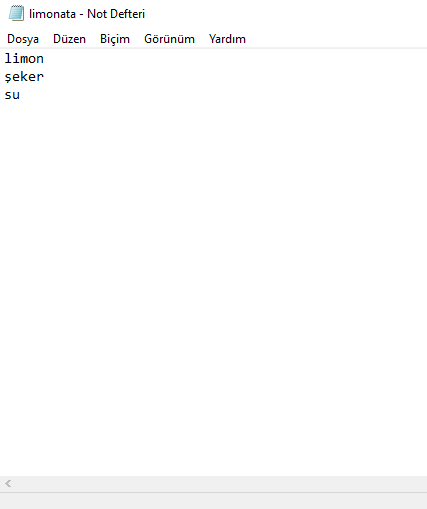
"git log" komutuyla bu yaptığımız commit i görebiliyoruz:



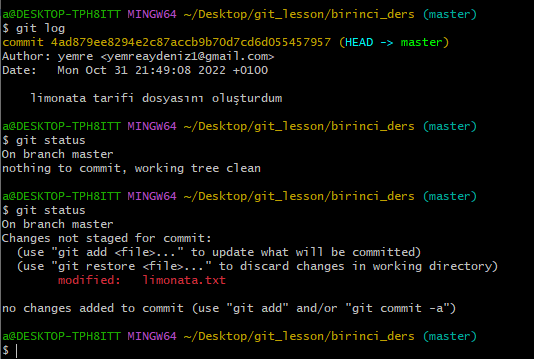
Şimdi "git status" komutunu girdiğimizde commit edilecek bir şey olmadığını söyleyecek. Çünkü biz dosyamızı repoya gönderdik:



Şimdi limonata dosyamızın içine girerek limonata için gerekli malzemeleri yazdık:

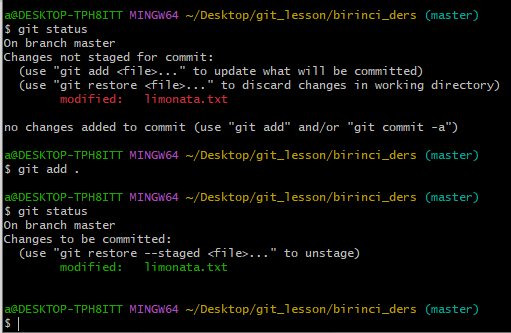


Dosyada bir değişiklik yaptıktan sonra git status yaptığımızda:

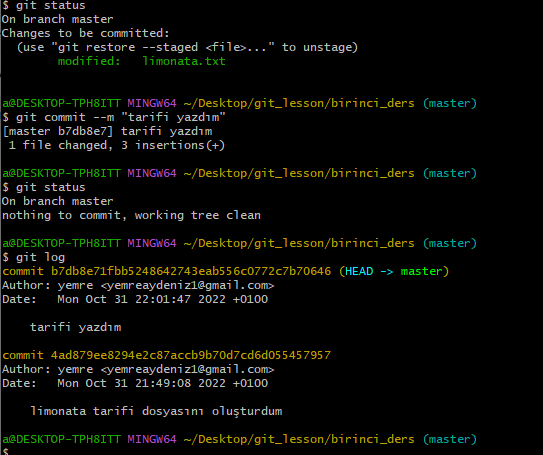


modified limonata.txt uyarısıyla dosyada bir değişiklik olduğunu ve commit olmadığını görüyoruz.

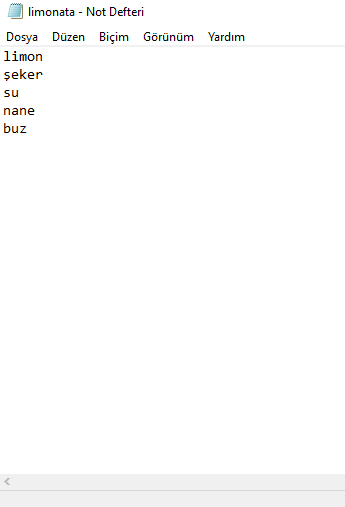
"git add ." komutuyla bütün değişiklikleri staging areaya ekliyoruz ve "git status" komutuyla kontrol ediyoruz:



"git commit - - m" komutuyla mesajımızı ekleyerek dosyayı commit ledir ve repoya gönderdik. "git log" komutuyla da iki versiyonu görüyoruz.



Sonra tarifimi geliştirmek istiyorum ve nane ile buz u ekliyorum:

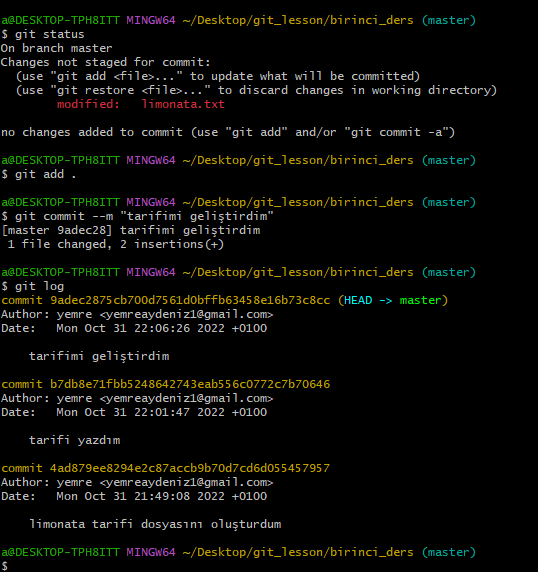


Şimdi yaptığımız değişikliği önce staging areaya sonra repoya göndereceğiz:

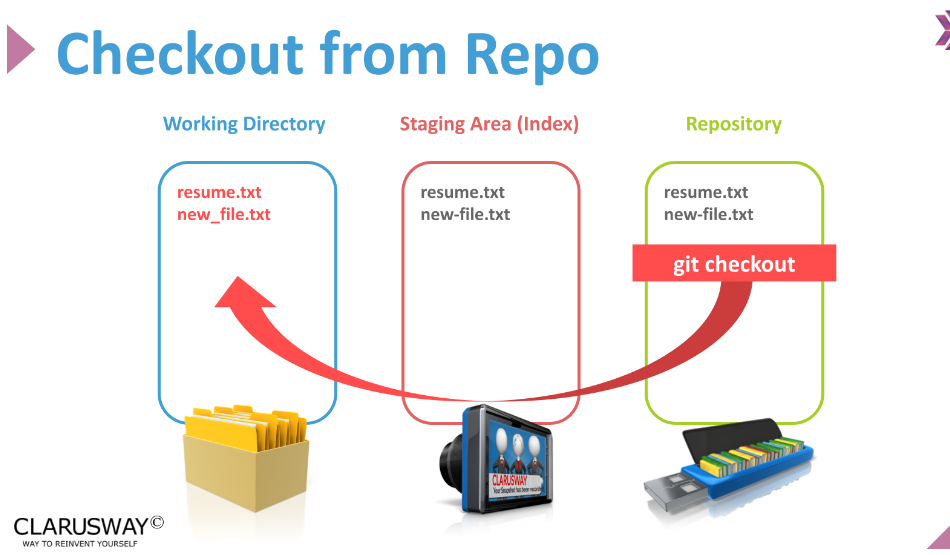
Sırasıyla;

"git status" ile yeni bir düzenleme olduğunu gördük.

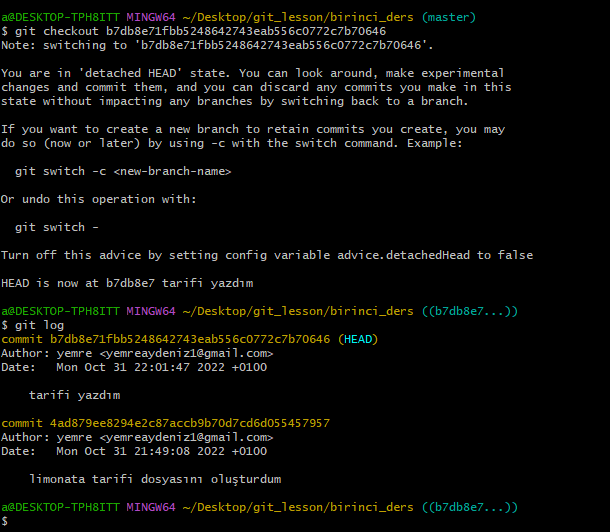
"git add ." Ve "git commit - -m" komutlarını yazıyoruz:



Versiyonlar arasında checkout komutuyla dolaşıyoruz:

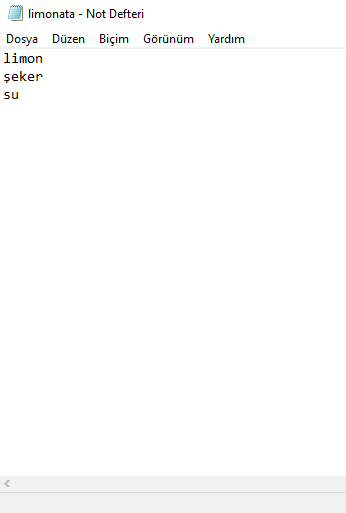


Şimdi checkout komutuyla daha önceki bir versiyona gideceğiz:



"checkout" komutuyla birlikte gitmek istediğimiz versiyonunda commit in yanında yazılı id yi yazdık ve o versiyona gittik. Sonra da "git log" komutuyla hangi versiyonda olduğumuzu tekrar gördük.

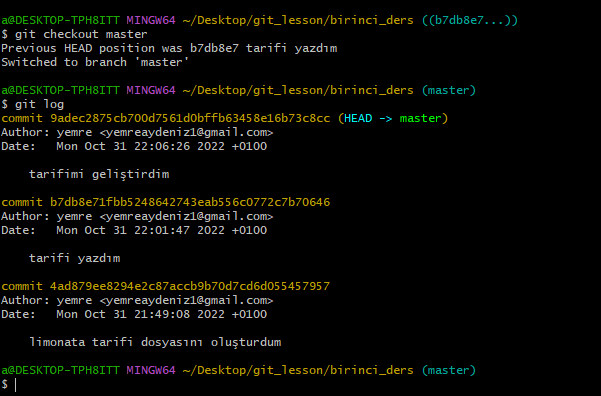
Şimdi limonata dosyamıza tekrar bakalım:



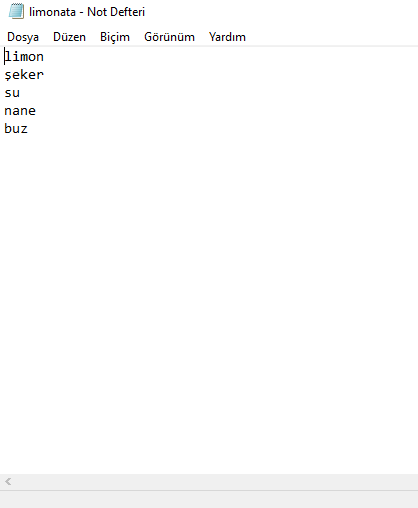
İkinci versiyona gittiğimiz için nane ve buz un silindiğini gördük.

"git checkout master" komutuyla da üçüncü yani en son versiyona geri dönebiliriz:

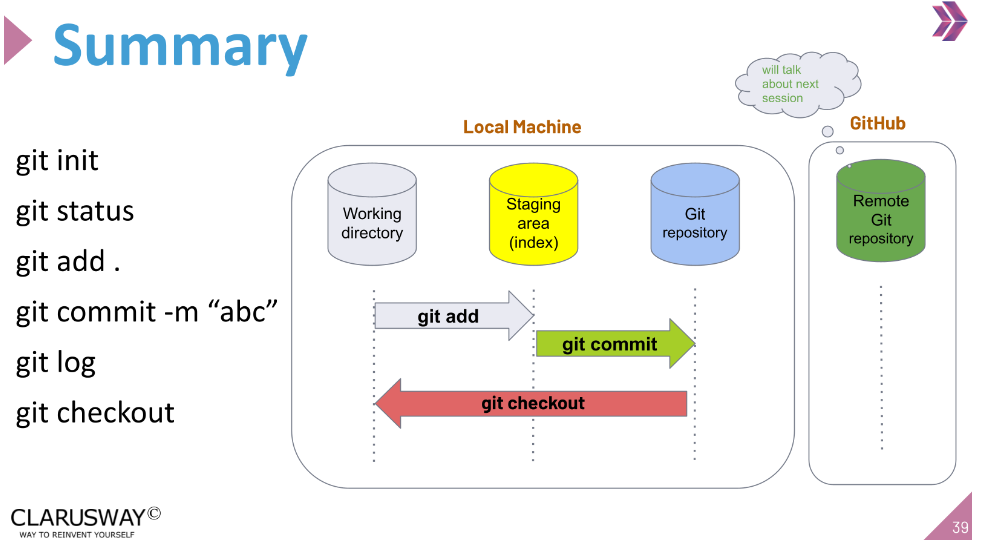
Ardından "git log" komutuyla da geri döndüğümüzü görelim:



Limonata dosyamızı açtığımızda en son versiyona geri döndüğümüzü yani nane ve buzun tekrar eklendiğini orada da görüyoruz:



Özetle:



.git klasörünü silersek bu klasör Git tarafından kontrol edilmez.

Ya da "ls -a" ve rm –rf .git" komutuyla terminalde silebiliriz.

Bir git klasörünün altında başka bir git klasörü olmaması lazım. Hata verir.

Git Bash kapandığında komutlar silinir. "history" komutuyla daha önce girdiğimiz komutları görebiliriz.