GIT Version Control System (Repository System)
Dosyları izler ve içlerinde yapılan değişiklikleri kontrol eder. Kim ne zaman neden yaptı?
İstanilen zamanda istenilen versiyona dönülmesini sağlar (recall)
Yanlışlıkla silinen dosyaları geri çağırılmasını sağlar (recovery)
Birkaç kişi aynı anda bir proje üzerinde çalışabilir (Can be used collaborately with others) (Team Work)
Local olarak kullanılabilir (Can be used locally) (No internet)
Comparing, Code Review, Blame (Kim yanlış yaptı?)
GIT BASIC COMMANDS
git --version : GIT versiyonunu öğrenmek için yazılır. (git version 2.31.0.windows.1)
rm -rf .git (Gizli dosyalar dahil tüm git dosyları silinir)
git config --list Tüm ayrıntıları gösterir. Senin git e kayıtlı username ve email bile görebilirsin
git config --gloabal user.name "Yaman" (İsmimizi git e tanıtır) (global kullanmazsab herbir repository için ayrı ayrı username tanımlayabilirsin ama gloabl olunca hepsine aynı username yazar)
git config --gloabal user.email "husoyaman1987@gmail.com" (Emailimizi git e tanıtır)
Bunları değiştirmek için aynı işlemi yaparak üzerine override edebiliriz
git config --gloabal core.editor "vim" (Tüm işlerimde vim editörünü aç demektir, nano da yapılabilirdi)
vim deneme.txt(Bu dosyayı vim editör ile aç demektir) ile içine girilir, i harfi ile INSERT mod açılmalı,birşeyler yazdıktan sonra ESC(Ki altta INSERT yazısı kaybolur)
Takiben :wq yapılarak kaydedip enter tuşuna basılarak çıkılır.
Birşeyler yazıp yine de çıkmak istiyor isek :q! işlemi yapılır ve enter tuşuna basılır
git config --global alias.co checkout (Bu şekilde komutlara kısa isim verebiliyoruz)
git config --global alias.br branch
git config --global alias.ci commit
git config --global alias.st status
git init ile git repository oluşturuyoruz (local)(HEAD oluşur ve .git dosyası gelir)
.git klasörünü rm -rf .git komutu ile silebilirsin (Muhtemel olacak hatalardan dolayı yapmak gerekebilir)
git branch komutu ile hangi branch ta (main mi master) olduğumuzu gösterir
git branch -m main ile master dan main e geçiş yapabiliyoruz
git help ile tüm git komutlarını görebiliriz
git merge --help ile merge ile ilgili bilgi alabilmek içi ilgili sayfaya gönderir (merge örnek)
git status git dosyamızın durumunu verir, tracklenmeyen değiştirilmiş klasörleri görürüz mesela
git clone (URL adresi) komutu ile github daki bir remote repositorydeki repo yu kendi bilgisyarımıza kopyalabiliyoruz (.git klasörü direk içinde geliyor)
git add . komutu ile bulunduğun dosyayı Staging Area ya atmak için kullanılır.Ayrı ayrı atılabilir.
Bu dosyalar takiben Repository e atılacaktır. Öncesinde bu dosyaların rengi kırmızı idi şimdi yeşil oldu.
git restore --staged deneme_4.txt (Commit etmeden staging area dan working directory e atmak için)
(git rm --cached deneme_4.txt komutu ile tekrardan staging areadan working areaya alıyoruz)
git commit -m (Commit the files on the stage) ile bunları Repository e atabiliyoruz. Değişiklikleri saklamak istersek mutlaka commit yapmamız gerekir
(Bu komutta ayrı ayrı atayım olayı yok)
git commit --amend -m "message" ile en son commit mesajını değiştirebiliriz (-m yazılmazsa vim editör açar,vim usuller standart)
git commit -am komutu ile modified dosyaları Staging Area yı pas geçip direk Repository e atabiliyoruz
git log ile Repository e atılan dosyaların tüm bilgilerini görebiliriz
git log --oneline komutu ile tek satırda gösterebilriz bilgilerini
git log --pretty=oneline
git log --graph (Görsellik katar tree gibi)
git checkout ID(İlk 5 tanesi yeterli) komutu ile önceki versiyonlara dönebiliyoruz. HEAD main den istenilen ID ye gelir
git checkout main komutu ile de yine sona devam edebiliriz. HEAD bir cursor dır ve nereyi gösteriyorsa oradayız(Şimdi main e tekrar geri geldi)
Committed (means that the data is safely stored in your local database.)
Modified (means that you have changed the file but have not committed it to your database (repo) yet.)
Staged (means that you have marked a modified file in its current version to go into your next commit snapshot.)
git remote add origin URL komutu ile local den github daki repositorye dosya atabilmek için ilişkilendirme işlemini yapıyoruz
(origin demek bizim remote repository branch demek)

git remote -v komutu (verify)(kontrol) ile projenin nereden nereye kopyalanacağını(import) gösterir.
Local repository min bağlantılı olduğu repository leri gösterir.
git push -u origin master/main komutu ile de projeni githuba göndermiş oluyoruz (-u upstreamin kısaltılması) (git push --set-upstream origin/main main komutu remote repositoryde yoksa)
git push https://github.com/Yaman35/Son.git gibi komutla daha kolay da atabiliriz.
(Eğer git push yaptığımız zaman göndermiyor hata veriyorsa başkası önceden değişiklik yapmıştır.Dolayısıyla senin değişiklikleri ilk önce git pull yapıp sonra push yapman gerekecektir)
git fetch komutu ile yapılan değişiklikleri gösterir.(Local repository ardından git diff main origin/main komutu ile farklılıkları görürüz)
git fetch -all (Fetch all the branches simultaneously)
git diff main origin/main (Benim localimdeki main branch ile origin(remote) main arasındaki farkı bana göster demektir
git diff <commit-id> <commit-id> İki commit arasındaki farklılıkları gösterir
git diff < branch 2> İki branch arasındaki farklılıkları gösterir
git merge origin/master komutu (git fetch) ile birlikte kullanılmalıdır.(to see the changes in working directory) Yapılan değişiklikleri bir nevi kabul ediyorum getir önüme (dosya içerisine import) demektir
Böylelikle yeni dosyalarla birlikte yapılmış olan değişiklikleri kabul etmiş, görmüş olduk
git pull origin main/master komutu ile de yukarıdaki iki komutun yerini tutar.(git pull için repository .git hatası verdiğinde git init yapmamız gerekir .git dosyası gelmesi için)
git pull ve git push işlemlerini aynı branchlarda yapar, birisi master birisi main ise olmaz mesela (remote-local)
git pull origin main ile direk github üzerinden çekiyoruz, değişiklik olmamsı gerekiyor yani commit yapmadan önce pull yapmalıyız !!!!!!!!!!1
(git pull origin main --allow-unrelated-histories komutu ile localdeki dosyamızı githuba atarken sıkıntı yaşarsak)
(git push origin main -f de zorla gönderir)
BRANCHES
git branch komutu benim bilgisyarımdaki local branchları görüntüleyebilirim. * işareti hangi branch üzerinde olduğumuzu gösterir
git branch <new branch> komutu ile yeni bir branch oluşturabiliyoruz takiben git branch komutu ile onu görebilriz bu sefer
git branch localdekiler
git branch -r remote dakiler
git branch -a hepsini gösterir
git branch -m komutu branch ismini değiştirmek için kullanılır
git checkout Yaman komutu ile branchlar arası switch işlemini gerçekleştirebiliyoruz
Burada ls yaparsak main deki dosyaların hepsini görürsün çünkü Yaman ı ordan oluşturmuştuk
git checkout main yaparsak tekrar maine döner
Yaman da iken bir dosya oluşturduk ve tekrar maine geçtiğimizde yaratılan dosya görülmedi bunun için merge yapılması gerekir, iki çeşit yöntem var
- Fast Forward Merge:
Hangi branch üzerine merge yapmak istiyorsak ilk önce ona geçmemiz lazım
Mesela main branch da iken git merge Yaman2 yaparsak, bu sefer Yaman2 deki dosyaları da görebiliriz
Sonra mesela Yaman branchına gittik ama Yaman 2 dekiler onda gözükmedi (Yaman2 branchı Yaman branchında iken oluşturulmasına rağmen)
Burada iken de şimdi git merge main veya git merge Yaman2 ile onları da göstertebiliriz
- Way Merge:
git checkout -b Yaman2 komutu ile yeni bir branch oluşturur ve hemen oraya gider (HEAD oraya gider), yukarıdaki kodda oluşturuyor lakin oraya gitmiyordu
İşte burada Yaman da iken oluşturduğumuz dosya gözükür bu sefer çünkü Yaman2 brachını Yaman branchında iken oluşturmuştuk
git branch -d Branch name Eğer bir branch bir başka brancha merge edilmişleri siler, edilmemişse onları silmez
git branch -D Branch name İster silinmek istenen branch başka bir branch ile merge edilmiş olsun, isterse olmasın her halükarda siler
git push --set-upstream origin New_Feature (Senin remote repository de New Feature diye bir branch yoksa sana bu şekilde tavsiye veriyor ve bununla yapabilliyorsun)
git log --graph --oneline Branchlar için tree komutu

NOT1!!!: git push yapınca sıkıntı çıkartıyorsa bil ki öncesinde senden değişiklikleri git pull ile çekmeni istiyordur
NOT2!!!: Eğer push etmek istediğin dosyanın branch ı remote repository de yoksa upstream ile orada yaratıp atabiliyoruz (git push --set-upstream origin new_branch)
NOT3!!!: Başkasının hesabından clone yapınca yeni bir branchda çalışmakta fayda var
Genel bir uygulama olarak başkasının hesabına push yapmadan evvel proje değişmiş olabilir
O yüzden onları da tekrar görebilmem için $ git remote add upstream(nickname)
https://github.com/tyler-clarusway/final-lesson