GİT NEDİR? NE DEĞİLDİR...

1- **Git** bir versiyon kontrol sistemidir.

2-Bu **açık kaynaklı** versiyon kontrol sistemi sayesinde; projelerimizin versiyon kontrol takibini yapabilir, dünya çapında projelere katkıda bulunabilir, kendi projelerimizi dünya ile paylaşabilir ve projelerimizin mobilitesini artırabiliriz.

3-Bu sistem sayesinde, bir projede aynı anda birden fazla kişi ile çalışabilir ve bir dosyada yapılan tüm değişiklikleri görüntüleyebiliriz.

4-Git != GitHub || GitLab

(Konu dışı: Windows nereden nereye geldi, en istikrarlı versiyonu XP idi, belki de şimdi kullandığımız Win.11 XP nin kodlamalarını içeren görsel desteklenmiş halidir? (Versiyonlar arası geçiş ile ilgili bir kanı, versiyonlama önemli, en son versiyon değil en istikrarlı-stabil- versiyon önemli.. Neyse kafanıza takmayın bunu, konumuz değil 😎)

BAŞLICA GİT TERİMLERİ

1-Repository

Bir diğer adıyla Kod Deposu. Tüm kodlarımızın tarihçelerinin barındırıldığı depodur. Proje dizinimizde git init diyerek bir repository oluşturabiliriz.

2-Working Directory

Kısaca Çalışma Alanı. Projemizin ana dizinidir. Bir dosyanın/klasörün Working Directory'de bulunması o dosyanın Git tarafından takip edildiği anlamına gelmez. Sadece Staging Area'daki dosyalar Git tarafından takip edilir.

3-Staging Area

Git tarafından takip edilen dosyalar burada bulunur. Gerçek manada fiziki bir alan değildir. Dosyalarımızın durumunu belirten hayali bir ortam denilebilir. Bu bölge Git' in sorumluluk alanıdır.

4-Commit

Kısaca Taahhüt veya Sözleşme diyebiliriz. Bir dosyada yaptığımız değişikliklerin kalıcı değişiklikler olduğunun taahhüdünü vermemiz, sözleşmeyi imzalamamız

gerekmektedir. Böyle ilgili dosya/dizinde yaptığımız değişiklikler repository'mize kaydedilir.

GİT CLI (KOMUT İSTEMCİSİ) TEST EDELİM

Git'i sistemimize kurduktan sonra, komut istemcimizde git --version komutunu çalıştırdığımızda bize sistemimizde yüklü olan Git'in versiyonunu döndürecektir. (Versiyonlarda farklılık olabilir. Herhangi bir versiyonu döndürmesi sizin için yeterlidir, Git'i başarılı bir şekilde kurulmuş demektir 😎)

git init komutu sayesinde mevcut dizinimizde bir Git Repository'si oluşturabiliriz ve sonrasında boş bir Git Reposu oluşturuldu şeklinde çıktı alırız.

Artık bu proje altındaki dosyalar Git ile takip edilebilirler. Fakat henüz takip edilmiyorlar❗ Bunun sebebini aslında yukarıda belirtmiştik. projemiz bizim Working Directory'miz yani çalışma alanımız. Fakat biz ne demiştik? Bir dosyanın veya klasörün Working Directory'de olması Git tarafından takip edildiği anlamına gelmez. Bu sebeple dosyalarımızı Staging Area'ya yani Git'in takip ettiği sanal ortama sözleşme/commit imzalayarak tanıtmamız gerekli. Bu kısa bilgiyi verdikten sonra bizim elimiz kolumuz olacak olan git status komutuna bakalım:

Git reposunu oluşturduğumuz dizin altında git status komutunu çalıştıralım. Bize repomuzun durumunu anlatırken aynı zamanda yapmak isteyeceğimiz işlemlerin komutlarını da hatırlatır 😇

Git "deneme" adlı dosyamızı gördü fakat henüz takip etmiyor. Bu sebeple Untracked files yani Takip edilmeyen dosyalar altında listeledi. Şimdi Git'e bu dosyayı takip et, yani Staging Area'ya al dememiz gerekiyor.

git add . veya git add dosya_adi komutunu kullandığımızda ilgili dosyayı Staging Area'ya ekler ve takip etmeye başlar.

Git artık dosyalarımızı takip ediyor fakat bu sefer de bize diyorki 'Bu dosyada değişiklikler var ama henüz commit edilmemiş'. Yani yaptığımız değişiklikler için sözleşme imzalamamızı istiyor. Bunu da git commit -m "ilk commit" komutu sayesinde yapabiliriz. git commit -m komutuna argüman olarak yolladığımız string bizim sözleşmemizin açıklama metni olacaktır. Log kayıtlarında bu şekilde görüntülenecek. Bizim ilk denememiz olduğu için "ilk commit" şeklinde bir açıklama yaptık, siz farklı bir açıklama yapabilirsiniz.

Tabii ki gerçek bir proje üzerinde çalışsaydık takım arkadaşlarımızın bilgilenmesi için yaptığımız değişikliği açıklayan nitelikte bir açıklama yazmamız daha uygun olurdu.

Eğer Git'i ilk defa kurduysak bu aşamada git bizden kendimizi tanıtmamızı isteyecektir. Bunun sebebi ise oluşturduğumuz commitleri (sözleşmeleri) bizim adımıza oluşturacak ve Repository üzerinde kimin değişiklik/güncelleme yaptığının bilgisini tutacaktır. Sistemimizde tek seferlik bu konfigürasyonu yapmamız gerekmekte. Gerçek bilgiler olması önemlidir ❗

git config --global user.name "ad soyad"

git config --global user.email "mail adresi"

Demiştik ki git status bizim elimiz kolumuz olacak. Şimdi hello.java dosyamızda rastgele değişiklik yapalım ve ardından git status komutunu çalıştıralım.

Gördüğümüz gibi Git bize hangi dosyalarda değişiklik olmuş, repomuzda neler olmuş bitmiş bunların raporunu vermekte. Buradan anladığımız kadarıyla da hello.java değiştirilmiş/güncellenmiş fakat henüz commit edilmemiş. git status komutu zaten bize ne yapmamız gerektiğini söylüyor. Biz git add komutunu zaten görmüştük.

PUSH - PULL NEDİR?

Push, yerel (local) makinadaki commit'leri uzak sunucuya göndermek (itmek) demektir. git push komutu sayesinde local commit'lerimizi uzak sunucuya gönderebiliriz. Aynı zamanda uzak sunucudaki commit'leri yerel (local) makinamıza çekmek için ise git pull komutunu kullanmamız gerekiyor.

GİTHUB İLE DEVAM EDELİM

-Eğer hali hazırda bir Git repomuz yok ise ilk bloktaki kodlara tabi olacağız,

-Local'de bir repomuz var fakat henüz bunu bir uzak sunucuya atmamışsak 2.bloğa tabi olacağız,

-Kodlarımız başka bir uzak sunucuda ise 3. bloğa tabi olacağız.

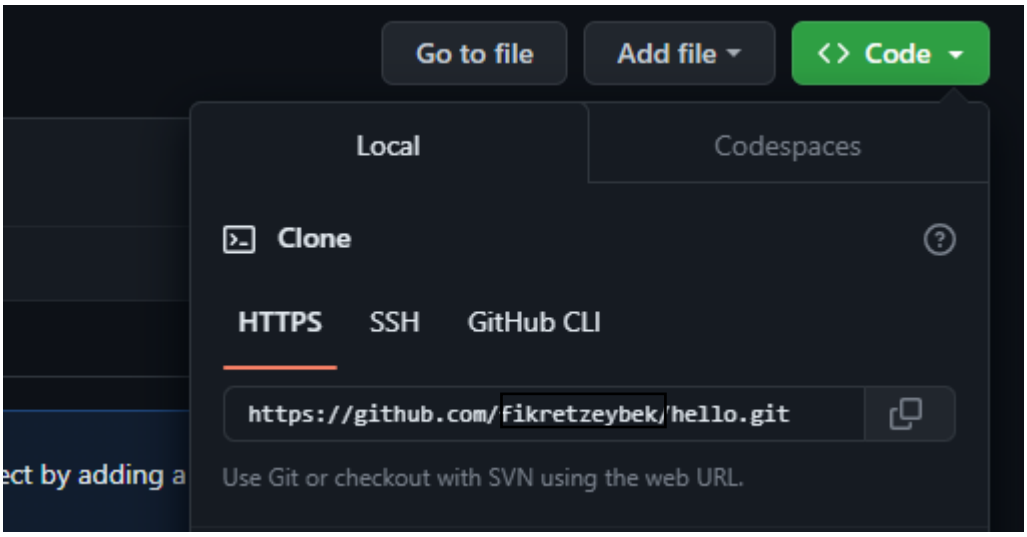
Şimdi bu komutları inceleyecek olursak;

git remote add origin -> origin adında bir uzak sunucuyu repomuza ekler, https://github.com/...../.....git -> uzak sunucumuzun adresidir.

git push -u -> local'deki commit'leri push et, origin -> repoma tanıttığım origin isimli sunucuya, master -> origin isimli sunucumdaki master branch'ine.

Not: Copy/Past işlemlerini tüm komutları tek seferde yaparak da gerçekleştirebilirsiniz.

CLONE NEDİR ?



Projelerimizi uzak sunucuya push ettikten sonra, bu repoyu o uzak sunucudan herhangi bir başka sisteme çekmek için git clone repo_adresi komutunu clone'lamak istediğimiz repo'nun adresini argüman olarak yollayarak kullanabiliriz. Ardından uzak sunucudaki repo'muz yerel (local) sistemimize kopyalanmış olacaktır. Bu işleme clone denmektedir.

(GitHub için clone adresine repo'muzun detayına girdikten sonra sağ üst köşeden erişebiliriz.) *-Derlenmiş Kaynak-*