**GIT KOMUTLARI**

---------------

**- 1.** **git init** --> Local repo oluşturmak için yani git ile klasörümüzün içindeki dosyaları

ilişkilendirmek için kullanılır

**- 2. git add .** --> Working space'deki (yani yerel) dosyadarımı staging area'ya(yani commitlemek için beklenen yer) gönderir

**- 3. git commit -m "mesaj"** --> Staging area'dan commit stora dosyalarımı göndermek için kullanılır.

(Commit yani version yani sürüm oluşmuş olur

**- 4. git push** --> Uzak repo'ya (GitHub) göndermek için kullandığımız kod.

Yalnız git push komutunu direk kullanmak istersek bir kere

git remote add origin - github'daki oluşturduğumuz repo'nun adresi

- git push -u origin master

-- Bu iki komutu tek seferde kullandıktan sonra diğer commit'lerden sonra sadece git push komutu kullanırız.

**--------Kurulum:**

1. Adim: Git uygulamasini indir ve kur (https://www.git-scm.com/)

2. Adim: GitHub hesap ve Repository olustur

3. Adim: Ornek bir proje olustur ve once Git e sonra GitHub a yedekle

**--------Komutlar**

git --version

git config --global user.email "email\_adresiniz"

git config --global user.name "isminiz"

git config --global -l --> Ayarları listeler

git init --> git ile ilişkilendirir

git status --> Working Space ve Staged deki değişiklikleri gösterir

git add . --> Working'den staging area'ya gönderir

git status

git diff --> Working space deki değişikliği gösterir

git diff --staged --> Staging Area daki değişiklikleri gösterir

git commit -m "first commit"

git show "hashcode" --> Versiyondaki değişiklikleri gösterir

git log --oneline

git remote add origin https://github.com/techproed2020/Git\_...

git push -u origin master

**- KULLANICI BILGILERI :**

git config --global user.name "kullanıcı adı / rumuz"

git config --global user.email "E-posta"

git config --list # Calisma ve kullanici bilgilerini göster

- YEREL DEPO :

git add <DOSYA(LAR)> # Yeni ve degismis dosyalari guncellenecekler listesine ekle

git add . # Yeni ve degismis dosyalarin tumunu guncellenecekler listesine ekle

git add -u # Silinmis ve degistirilmis dosyalari guncellenecekler listesine ekle

git rm <DOSYA(LAR)> # Calisma agacında ve dizinde dosyalari kaldir

git rm -f # Calisma agacında ve dizinde dosyalari zorla kaldir

git commit -m 'not' # Değisiklikleri depoya kaydet

git commit -a -m "not" # Tum değisiklikleri depoya kaydet

cat .gitignore # Dosyayi depoya ekleme

git rm --cached <DOSYA> # Dosyayi takip etmeyi birak

git diff # Degisiklikler arasindaki farklari goster

git diff --cached # Listeye Eklenen Değişiklikler Arasındaki Farkları Göster

git status # Calisma agacindaki durumu goster

git log # Islem gunlugunu goster

**- UZAK DEPO :**

**git clone <ADRES>** # Uzaktaki depoyu klonla

**git pull** # Depodaki son degisiklikleri al

**git push** # Yereldeki degisiklikleri uzak depoda uygula (origin master)komutuda eklenebilir

**- DAL (BRANCH) KOMUTLARI :**

git branch <DAL ADI> # Dal olustur

git branch # Dallari goster

git checkout <DAL ADI> # Calisilan dali degistir

git merge <DAL ADI> # Dallari birlestir

git branch -d <DAL ADI> # Dal sil

**- DIGER KOMUTLAR :**

git --version # Git versiyon numarasını göster

git --help # Git yardım sayfasını göster

git remote -v # Uzak depo adresini ver

git log --since=<LIMIT> # Iki zaman araligindaki commitleri goster

git shortlog -s # Commit yapanlarin isim ve commit sayilarini goster

git shortlog -e # Commit yapanlarin isim ve E-postalarini goster

git shortlog -n # Commit yapanlari commit sayisina gore sirala

git reset -- hard HEAD # Son yapılan degisiklikleri iptal ederek HEAD geri don

git checkout -- <DOSYA> # Sadece bir dosyayi depodaki haline geri getir

git revert HEAD # Son yapilan commiti geri al

git stash # Commit yapilmamis degisiklikleri kaydet

git stash pop # Commit yapilmamis degisikliklere geri don

git stash list # Commit yapilmamis degisiklikleri listele

git stash drop # Commit yapilmamis degisiklikleri kaldır