

Yukarıdaki config içerikli kodlar git kurulumu yapıldıktan sonra ilk aşamada ilgili localin github ile ilişkilendirilmesi için kullanılıyor.

Github çok geniştir hepsini bilmemize imkan yoktur işimize yarayan kısmını öğrenmemiz yeterlidir

- Bu videoa 4 temel senaryo gorecegiz ve burada kullanacagimiz git kod sayi -si 10 u gecmez hemde

ezberlemeye gerek kalmadan

- sirketlerde bu tarz surekli geliştirme programlari kullanildigindan bilinmesinde fayda var

- Github yada türevi bir program kullanmak istiyorsanız bilgisayarınıza mutlaka git kurmak zorundasiniz

- İşi temeli git; Github ve benzeri(Gitlab vb) programlar sadece gitin repolarini saklamak icin kullanilan

bulut tabanli bir veri saklama platformudur

-- Özetle işi yapan "Git" veriyi saklayan "Github"dır

- Github Git adli bir VCS (Version Control System) barindiran bulut tabanli bir hizmettir

- Yerel bilgisayardaki klasorlerle (workstation), uzak veri depolama (Remote Repository yani Github)

arasinda araclik yapan "git"'dir

- Git bireysel kuck veya kapsamli buyuk projelerde herseyi hizli ve verimli yonetmek icin tasarlanmis,

ucretsiz ve acik kaybakli "dagitilmis versiyon kontrol sistemidir" (distibuted version control system)

-- dagitilmistan maksat verileri tumunun sadece yerl bilgisayarlarda veya sadece bulut tabanli dedigimiz

sunucularda(Github gibi) degil; hem yerel bilgisayrlarda hemde bulut sunucularda belirli kurallar

dahilinde depolanmasidir. Bu durm bize verinin kaybolmadan saklanmasi ve birden fazla (baya fazla)

kullanicinin ayni proje uzerinde beraber, ayri ayri calisabilmesi de dahil olmak uzere bircok imkan

ve kolaylik saglar.

- Git ile yapilan herbir senkronize et emrine "version" denir. Versiyon, projemizin yedeklenmis herbir adimidir

- Git programini diger muadillerinden ayiran en onemli ozelligi "branch" sistemiyle birden fazla kisi ve/veya grubun

birlikte calismasini saglayacak sekilde VCS(Versiyon <Kontrol Sistemi) olusturmasidir

- Git push ==> Lokaldeki kodlar uzak repo'ya gider

- Git pull <== Uzak repo'daki kodlar lokale gelir

- Git ve Github sadece kodlarimizi degil herhangi bir klasorun yedelenmesindede kullanilabilir

- version olusturuldugunda projenin tamamini degil sadece degisiklikleri, ve eski hali nasilsa onu kaydeder

- it klasor tabanli calisir klasorun icerigine bakmaz ne var ise onu yedekler & degisiklikleri kaydeder

- bir klasorun git ile irtibatli olup olmadigini anlamak icin klasorun icinde

.git gizli klasorunu gormemiz gerekir. Bu git klasoru terminalde git init komutunu ile olusturulur

-- bunun icin bilgisayarimizin gizli ogelri gosterme secenginin aktif olmasi gerekir

buradaki .git klasoru silinirse yereldeki yedeklemelerde(yapilan versionlama vb.) silinmis olur

-- bu .git klasoru silinerek ilgili klasorun git ile iliskisi kesilmis olur

- Git hali hazirda MsDos sistekiyle, yani cmd-siyah ekran, ile calisir. İnteliJ'deki hali hizli erisim icindir

yani cmd ile InteliJ'deki terminal ayni seydir

- Git'i kullanmak icin butun komutlari bilmemize gerek yok. ihtiyacimiz kadarini OGRENMEMIZ yeterli

- git init : ilgili klasor veya proje git ile irtibatlandirilir

- git add . : klasorde yapilan tum degisiklikler git'e eklenir (ilk defa yazilirsa hepsi eklenir)

nokta yerine ilgili dosya adi yazip sadece o dosya icin commit olusturabiliriz

- git commit -m "isim" : versiyon olusturur ve versiyon'a isim ismini verir

- git ile Github'a dosya eklemek icin;

-oturum acilmis olan Github sitesinde yeni repo olusturulur, sonra sitedeki url ile olusturulan

repository sayfasinda izah edildigi sekilde komutlar terminale girilerek islem tamamlanir

\*- Bir klasoru sisteme yuklemek istedigimizde klasorun adres satirina cmd yazip eterladigimizde MSDos

terminali (siyah ekran) acilir ve intelij terminalindeki islemlerin aynisi burada tekrarlanir

- Github'dan dosya cekme;

1- once bir klasor olusturulur ve git init yapilir

2- ilgili repo linki Github adresinden kopyalanir (code'ye gel cikan adresi kopyala)

3- git pull kopyalanan url yazilarak islem tamamlanir

- Git hubdan cekilen dosya ile git ve Github'i iliskilendirme;

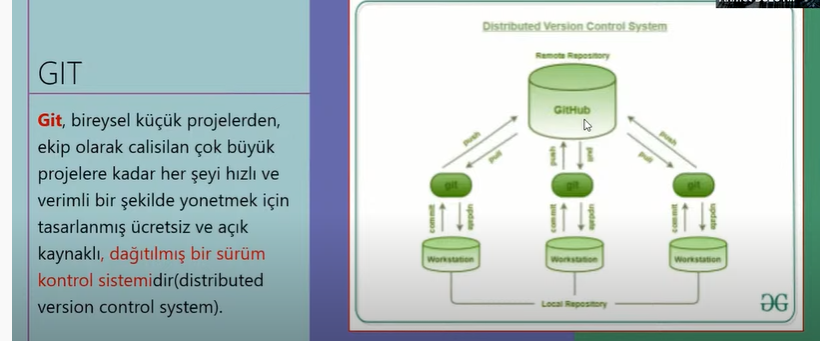
1- Klasor acilarak klasor adres satirina cmd yazilarak terminale girilir

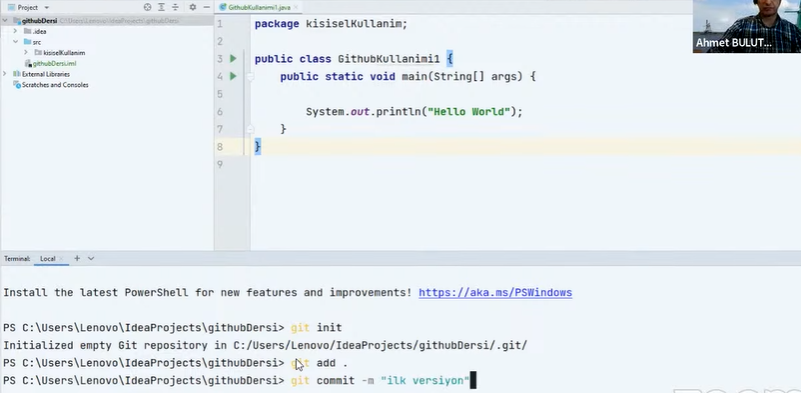
2-git branch -M main ile remote repodaki branch tanimlanir

3- git remote add origin "https://github.com/repoPath" komutu ile remote ve lokal repo baglanir

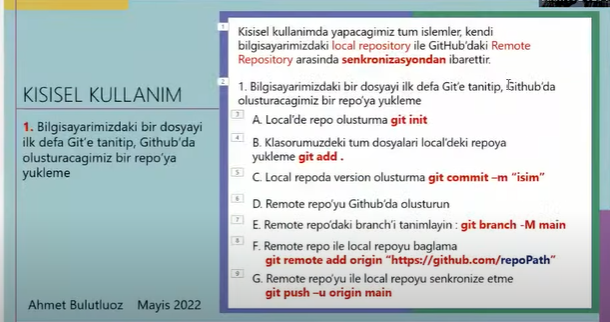
4- git push -u origin main ile remote repo ile lokal repo senkronize edilir

-- daha sonra yukarida yukaridaki gibi git push islemi yapilir

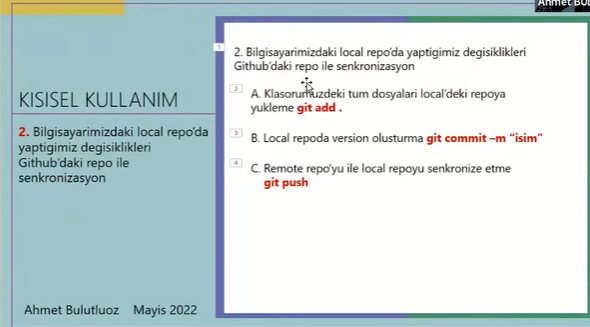




Bu 3 komut ile localdeki görevimizi bitirmiş oluyoruz. İkinci aşamada görevimiz remote sunucudaki işlemleri yapmak. Bu komutları inteelij üzerinden yazabildiğimiz gibi komut satırı üzerinden de yazabiliriz.



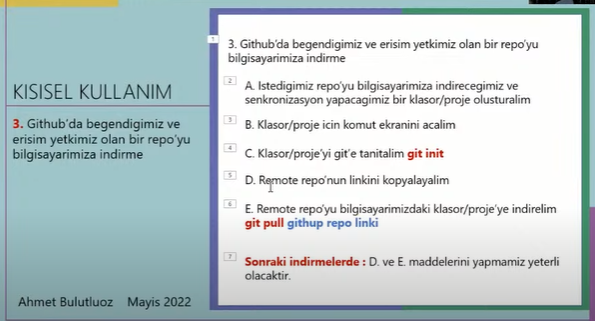
Yukarıdaki işlemlerle sıfırdan bir projeyi localde git ile ilişkilendirip daha sonra remote sunucudaki işlemlerle githuba gönderiyoruz.



Artık “init” ile dosyayı başlatmaya gerek yok zaten projemiz, dosyamız git ile ilişkilendirilmiş.

Amacımız projedeki değişiklikleri uzaktaki sunucuya göndermek. O nedenle yapacağımız tek şey yukarıdaki komutlarla terminalde ilgili işlemleri yapmak. Burada önemli olan “commit” ile versionları oluşturmak. Örn : dosyadaki bir değişikliklik için “ikinci değişiklik”vb. yazılabilir.

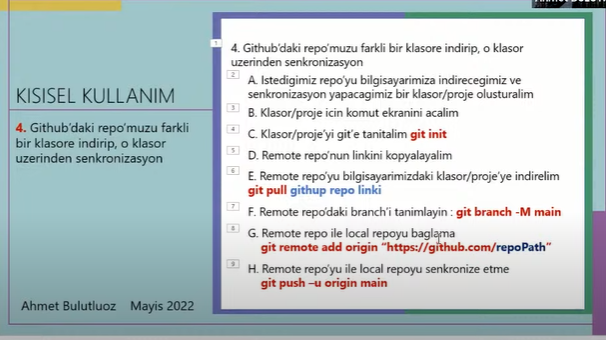
Dosya barındıran klasörler için ise sağ tuş ile o klasörün pathini bularak cd ile komut satırından klasöre giriş yaparız ve yukarıdaki komutları uygulayarak klasör içindeki değişiklikleri göndermiş oluruz.



Bu adımda ise beğendiğimiz bir repoyu pc ye indirebiliriz. komut satırında yukarıdaki işlemleri yaparak kolayca dosyaları çekebiliriz. En başta kendi localimizde bir klasör oluşturup o klasörü git ile irtibatlandırırız. ( git init ). Daha sonra indireceğimiz reponun linkini kopyalayarak git pull LİNK komutunu başlatırız. Ve tüm dosyaları , kodları kendimize alabiliriz.

4. Adım

Bu adımda ise farklı bir senaryo olacak. Şöyle ki daha önce git ile ilişkilendirdiğimiz devam ettiğimiz bir proje veya dosya pc mizden silinmiş olabilir , kaldırmış olabiliriz ve bu durumda localde kaybolan dosyayı github dan yeniden locale linki kopyalayarak pull etmiş (çekmiş) olabiliriz. Daha sonra çektiğimiz bu dosya üzerinden projemize devam edip çalışırken yaptığımız değişiklikleri 2.adımdaki gibi push edemeyiz. Her ne kadar çektiğimiz dosya git ile irtibatlı olsa da sıfırdan bir projenin git e tanıtılması gibi bir süreç olmamıştır , pull edilerek oluşmuştur. O nedenle bu dosyadaki değişiklikleri farklı bir şekilde uzak repoya göndeririz.



Pull ettikten sonra sanki sıfırdan bir projenin githuba gönderilmesi sürecinde remote taraftaki işlemlerin yapılması gerekir. Yani F,G,H daki komutların çalıştırılması gerekir.