**GİT**

Sürüm kontrolü yapıyor. Yani yaptığımız yazılımın aşamalarını kaydediyor ve bunları saklıyor. Yazılım başlangıcından bitimine kadar olan bütün aşamaları saklıyor. Kodun çalışma kopyasını depo olarak kullanıyor. Yazılımda herhangi bir değişim olmuşsa ekleme ya da çıkarma yapılmışsa hepsini hafızasına kaydediyor ve ilk haliyle son hali arasındaki değişimi görebiliyoruz . Değişikliğin kimin tarafından ne zaman yapıldıgınıda kaydediyor. Yanlışlık yada silme durumunda geri getirmemize olanak sağlıyor. Git dağınık sürüm kontrol sistemi kullanıyor; yani yedekleri tek bir sunucuya atmıyor diğer bilgisayar lardada yedekler dosya şeklinde tutuluyor böylece herhangi bir veri kaybı yaşanmıyor ve farklı yerlerdeki yazılımcılar dan da yardım almamıza olanak sağlıyor. Git i diğer sürüm kontrol sistemlerinden daha kullanışlı hale getiren özelliği yazılımda ekleme çıkarma vb. değişim durumlarında sadece değişen kısmı alıyor ek olarak diğerinin üstüne ekliyor. Değişmeyenleri ellemiyor. Bütün verileri kendi yerel belleğinde tuttuğu içinde hız olarak katkı sağlıyor. Açık kaynak yazılımdır. İsme değil içeriğe önem verir ve Listelemeyi ona göre yapar. Güvenliğe çok önem veriyor. SHA1kullanıyor. Çakışmaları önlüyor. SHA1 sayı sistemi kullanarak şifreleme yapıyor. SHA1 le sadece şifreleme yapabiliriz. Şifre çözümlemeye izin vermez. MD5 algoritması kullanarak özetler oluşturur böylece çakışmaları önlemiş olur. MD5 tek yönlü şifreleme yapar. Verinin boyutundan farklı olarak kendi 128 bitlik özetini çıkarır. Kurulum her işletim sistemine göre farklılık gösteriyor. Kurulumdan sonra 3 farklı git gelir. Hepsi git uygulamasıdır ama ara yüzleri farklıdır. Kurulumdan sonra bizden istediklerini yazıyoruz ayarlamaları yapıyoruz. Git sistemine kaydetmek için ismimizi epostamızı ister bunun nedeni başka yazılımcılarla çalışırken neyi kimin yaptığını bilmektir. Kurulumdan sonra git i kullanabiliriz. Git te her kodun başına git uygulaması olduğunu belirtmek için git yazarız. Git te projelerimiz git deposu içinde saklanır. Dosyalarımız git deposuna yollanmadan önce aşamalardan geçer yaptığımız işlem ilk önce gösterilir kesin etki etmesi için onu git add komutuyla geçiş bölgesine atmamız gerekir daha sonra git commitle git deposuna göndeririz. Commit git te bir önceki bellek kopyasıdır. Projede yaptığımız değişiklikleri kodlarla takip edebiliriz. Git status komutuyla yazılımda meydana gelen değişikleri görebiliriz. Herhangi bir dosyayı sildiysek, ekleme yaptıysak, dosya taşıdıysak vb. işlemlerde git status yaptığımızda bize yapılan değişikleri kırmızı renkle gösteriyor ve yapmak istediğimiz benzer işlemleride su kodlarla yapabilirsiniz diye örnek olarak yazıyor. Git le projemizde istemediğimiz dosyaları silebilir dosyaları isimlendirebilir başka bir dizin veya dosyaya taşıyabiliriz. Eğer çalışmamızın herhangi bir yerinde yanlışlık yapmış ve kaydetmişsek bunuda geri alabiliriz. Çalışmalarımızı kaydettiğimizde karışmasını önlemek istersek projelerimizi etiketleyebiliriz versiyon1 gibi veya tarih yazarak. Böylece daha önceki hallerine geri dönmek istediğimizde daha rahat arama yapabiliriz ve drek etiket adını yazarak var olan haliyle değiştirebiliriz. Git te uzak uçbirimlerle çalışabiliriz yani projemizi uzaktaki yazılımcılarla paylaşabilir onların paylaşılmış projelerini görebiliriz. Bu sayede hem kendi projemizde onların yardımlarını almış oluruz hem de onların projelerine yardım ederek farklı kod mantıklarını da görmüş oluruz. Aynı zamanda ortak projede yapabiliriz. Uzak uçbirimlerdekilere verileri push ederiz yani yapılan değişikleri uzak sunucuya göndeririz böylece kodun son hali üzerinden yazılımımızı görebilir ve ona göre fikir üretebilirler. Aynı şekilde uzak uçbirimlerdekilerin yazılım veya verilerinide pull yöntemiyle alabiliriz. Uzak uçbirimlerle çalışırken her iki tarafta yazılım üzerinde değişiklik yapacağından sıkıntı çıkabilir. Böyle durumlarda Git te branch kullanarak bunu çözebiliriz. Branch ta ana yazılıma bulaşmadan ek yazılım dalları açılır ve bu dalda bir sorunu giderebilir ya da yazılımın eksik bi kısmını tamamlayabiliriz. Gitte varsayılan ana dal master dalıdır. Eğer yarattığınız branch taki kodlar yazılıma gerekliyse ya da yazılımda işimize yarayacaksa vb. durumlarda branch la master ı ya merge yöntemiyle yada rebase yöntemiyle birleştiririz. Merge yönteminde yarattığımız branch ı master branchına entegre ederiz. Rebase yönteminde ise değişikliğin ortaya çıktığı yamayı alıp ortak ata üzerinde birleştirebiliriz. Git te projelerimizi uzak depolarda tutmak için uzak depolar vardır. Bunlardan bazıları Github, GİtLab ve Bİtbucket dir. Github en çok kullanılandır. Projemizi herkese açık olarak koyarak başkalarının projerimiz hakkındaki düşüncelerini öğrenebilir ve başkalarının projelerini takip edebiliriz. GitHub a projelerimizi git üzerinden kodlarlada koyabiliriz. GitLab Github ve bitbucketle aynı işlevi görür bitbuckete göre arayüzü daha zengindir. Bitbucket Git in Bitbucket ile birlikte çalışmalarını sağlamak için bazı düzenlemeler yapmamız gerek. Bİtbucekette dosya oluşturmak, git te olan dosyaları bitbuckete çekme kopyalama işlemleri yapmalıyız. Bitbuckette dosya grupları oluşturabilir paylaşım yapabiliriz ancak dosyaları silme ve orjinalini değiştirme sadece grup sa yöneticiye tekse kullanıcıya verilmiştir.