Sürüm Kontrol Sistemleri

Alp Altan 210601068
alpaltnceng@gmail.com

Sürüm Kontrol Sistemleri nedir? (Version Control System)

Bir yazılım projesini, bir dokümanı veya bir dosya üzerinde yaptığımız değişiklikleri kaydeden, bunları kendi bünyesinde kronolojik olarak sıralayan düzenleyici ve yedekleyici bir sistemdir. Günümüzde bir çok kişinin işbirliği halinde çalışması gereken yazılım projelerinde sıklıkla kullanılmaktadır. İşbirliği halinde olan kullanıcılar arasındaki bağlantıyı sağlamak bir projenin gidişatı için çok önemlidir eğer VCS olmasaydı proje teslim tarihleri sarkabilir, modüller arası çakışmalar olabilir, dosyalar karışabilir/bozulabilir vb.

Sürüm Kontrol Sistemi çeşitleri?

VCS ilk kullanılmaya başlandığından bu yana gelişen teknoloji ve ihtiyaçlar neticesinde pek çok değişime uğradı. İnternet erişimi eski zamanlarda bu kadar kolay ve çok olmadığı için ilk başlarda Local Version Control (Lokal Sürüm Kontrol) dediğimiz tek bir bilgisayar üzerine kurulmuş sürüm kontrol sistemi olarak ortaya çıkmıştır.

Lokal Sürüm Kontrol sisteminde kullanıcı kendi bilgisayarını sunucu gibi kullanıp, bütün dosyalarını statik olarak kendi cihazında depolar. Commitlerimizin her biri ayrı bir sürüm olarak tutulur ve çözülmesi zor bir depolama sorunuyla karşılaşırız bunun dışında bir süre sonra okuması ve anlaması zorlaşacak kadar çok dosya içinden aradığımızı bulmak işimizi kesinlikle zorlaştıracaktır. Bu eksiklikleri düzeltmek ve daha işlevsel çalışmasını sağlamak için ise Centralized Version Control (Merkezi sürüm kontrol) dediğimiz yapı ortaya çıkmıştır.

Merkezi Sürüm Kontrol sistemlerinde ana yapı repository dediğimiz depomuzda saklanır. Ana yapımız bu aşamada; bir yazılım projesi, bir doküman, ortak geliştirilen bir websitesi olabilir. MSK sisteminde ana yapı harici bir sunucu gibi davranır ve kurulduğu bilgisayar üzerinden Admin yetkisine sahip olarak ilerideki kararları ona göre verir. MSK sisteminde kullanıcılar Ana yapıya bağlanır ve yaptıkları değişiklikleri commit olarak sunucuya iletiriler. Sunucuda admin yetkisi olan kullanıcı ona gelen değişiklikleri onaylar ya da onaylamaz bu aşamadan sonra sunucunun kendi bilgisayarındaki ana dosyanın değişmesi diğer kullanıcıları da ilgilendireceği için bazen karışıklığa yol açmaktadır. MSK sistemlerindeki en büyük eksiklik bütün veri tabanının tek bir cihaz üzerinde depolanmasıydı. Sunucu bilgisayarına erişim kaybedildiğinde diğer kullanıcıların da değişiklik yapması pek mümkün olmuyordu bu sebepten geliştirilen ve günümüzde halen yaygın olarak kullanılan Distributed Version Control (Dağıtık Sürüm Kontrol) sistemleri ortaya çıkmıştır.

Dağıtık Sürüm Kontrol sistemleri günümüzde oldukça yaygın ve işlevsel olarak devam etmektedir. Temelindeki mantık tek bir sunucuyu her kullanıcının kendi cihazına kopyalamak yatar bu sayede her kullanıcı ana repositoryi kendi makinasına indirmiş ve sunucuya bağlı kalmadan geliştirmelerine devam edebilecekti. Sunucu makinasına bir zarar gelmesi halinde diğer makinaların birinden en temel hali tekrardan elde edilebilecekti bu sayede bütün cihazlardan dosyalar silinmediği takdirde hiçbir zaman yedek sorunu yaşanmıyordu.

Kullanıcılar offline olarak geliştirdikleri dosyaları commit attıklarında tekrardan bir admin onayından sonra ana sunucuya onaylanan değişiklik eklenmiş oluyor. Kullanıcıların kendi bilgisayarlarına kopyaladıkları depo, ana depodan daha eski bir sürümde kalırsa program hata verebilir. Bu sebepten ötürü her kullanıcı düzenli olarak ana repodan verileri çeker (pull eder) ve kendi makinasındaki repoyu güncellemiş olur.

Kullanılan Sürüm Kontrol Sistemleri nelerdir?

Günümüzde GIT, Github, GitLab, BitBucket, Mercurial, PerForce, AWS Code Commit, Microsoft TFS gibi bir çok sürüm kontrol sistemi mevcut. Her firma ya da ekip kendine en uygun gelen Sürüm Kontrol sistemini seçer ve proje başında geliştirmelere öyle devam edilir. Yukarıdaki versiyon kontrol sistemlerinden en popüler olanları Git tabanlı olan Github ve GitLabdır.

Öncelikle GIT ne işe yarar ona bakalım. Git, bir yazılım projesi geliştirirken kullandığımız sürüm kontrol sistemidir. Git sayesinde repomuzda yaptığımız her değişikliği kaydedebilir, olası bir hatadan sonra durum çözülemeyecek derecede derinleşmeden ve kötüleşmeden geriye dönüp hatasız kısımdan devam edebilmemizi sağlar. Git ücretsiz ve herkesin kullanımına açık bir platformdur bu sayede sürekli gelişen ve kendi kendini oldukça hızlı büyüten bir topluluğa sahip. Bugün GitHuba girip biraz arama yaptığınızda eminim ki bulamayacağınız şey yok. Ayrıca “Vay be, bunu da mı yapmışlar!” dediğinizi duyar gibiyim.

GitHub, GIT tabanlı bir sürüm kontrol sistemidir. Sürümler arası geçişler oldukça kolay ve öğrenmesi basit bir programdır. Son yıllarda Desktop olarak da uygulaması geliştirilen GitHub, kullanması gittikçe kolaylaşan bir Sürüm Kontrol Sistemidir. GitHub kullanmamız için yapmamız gereken 2 şey var. Bunlardan ilki bir hesap açmak ve ikincisi ise bir bilgisayara sahip olmak. Windows, MacOS ya da Linux hiç fark etmez bir bilgisayarımız olsun yeterli. Öncelikle Git programını bilgisayarımıza kuruyoruz sonrasında git komutlarını çalıştırıp çalıştıramadığımızı kontrol ediyoruz. Eğer git komutlarını bilgisayarımız algılıyorsa GIT başarıyla bilgisayarımıza kurulmuş olacaktır. Sonrasında ise bilgisayarımıza bizim GitHub profilimizi tanıtmamız ve pull, push, clone, commit vb.. işlemlerde Git sisteminin bizim hesabımızı tanıyıp ona göre işlem yapması gerekir. Ben public olarak paylaştığım bir repoyu yeni aldığım bilgisayarımda clone edip kendi bilgisayarıma tekrar yüklediğimde yaptığım değişikleri ana sunucuya yönlendiremediğimi görürüm. Bu aşamada nerede hata yaptığımızı anlamak güç değil herhalde? Cihazımıza GIT sistemi kullanarak bir anahtar oluşturuyoruz sonrasında ise GitHub hesabımızdan bu anahtarı kendi profilimize kaydediyoruz bu sayede o anahtarın kayıtlı olduğu bilgisayar üzerinden gelen push, pull, clone, commit gibi istekleri artık tanıyabilecek ve yetkilimize erişebileceğiz.

GitLab, GitHub alternatifi olarak daha yeni çıkmış sayılabilecek bir Sürüm Kontrol Sistemidir. Yine GIT altyapısını kullanan GitLab web üzerinden bize daha geniş bir yelpaze sunar. Bazı şeyleri daha kolay yapmamızı sağlar GitHuba göre daha pratiktir.

GIT altyapısına sahip bu iki sürüm kontrol sisteminin rakiplerinden öne çıkmasını sağlayan en büyük özelliklerinden birisi ise Branch dediğimiz dallanma yöntemi. Kullanıcı kendi makinasına kopyaladığı depoda değişiklik yaparken bunları ayrı bir daldan yürütüp en son admin onayıyla birlikte ana dala güncellemek isteyebilir. Bir sosyal medya uygulaması geliştirdiğimizi ve 4 kişilik bir ekip olduğumuzu varsayalım. A kişisi uygulamanın main feed dediğimiz akışı ile ilgilensin, B kişi uygulamanın sohbet kısmıyla ilgilensin, C kişisi profil sayfalarıyla ilgilensin, D kişisi ise topluluk sayfalarıyla ilgilensin.

Bu 4 kişinin bağlı oldukları bir proje yöneticisi olduğunu varsayalım. Eğer dallanma sistemini kullanmıyor olsaydık, her kullanıcı geliştirmelerini tam olarak bitirmeden commit attığında ve pull request dediğimiz proje yöneticisinin onayına sunduklarında, onaydan geçen her kod direkt olarak orijinal kodumuza geçecektir. Projenin sonunda doğru gelinip bir hatayla karşılaşıldığında ise çok eskilere gidip o hatayı bulmamız bizim için karmaşıklaşacaktır. Dallanma sistemi ile birlikte A kişisi Main Feed, B kişisi Chat, C kişisi Profil Sayfası, D kişisi ise Topluluk sayfalarını geliştirirken ayrı ayrı dallarda sadece kendi değişiklikleri ile sorumlu olarak geliştirme sağlaycaklardır.

MacOS GitHub, GitLab kullanımı ve avantajları/dezavantajları nelerdir?

Ülkemizde MacOS işletim sistemini kullanan cihazlar oldukça az sayıda. Windows cihazlara ulaşımın daha kolay ve yaygın olmasından dolayı insanımızda MacOS işletim sistemine karşı bir ön yargı mevcut. MacOs işletim sistemi terminal kullanımı açısından Windows cihazların bir hayli önüne geçmiş durumda. Eğer sadece mail atmak ya da slayt düzenlemek için kullanmıyorsanız terminale en az bir kez işiniz düşmüştür. Windows cihazlara nazaran çok pratik ve işlerimizi kolaylaştıran kısa yolları bulunmasına rağmen öğrenmesi ve alışması zaman alacak bir sistemi mevcut Mac cihazların. GitHub ve GitLab kullanımı açısından da öyle. GitHub ve Gitlab rakiplerine göre hızlı ve verimli, çevrimdışı çalışma imkanı sunan, dallanma sayesinde düzenli olmasıyla, geniş bir topluluğa sahip olmasıyla rakipleriden öne geçiyor ancak bunlara karşılık ilk başlayanlar için öğrenmesi zor ve kişi sayısının çok arttığı projelerde conflict dediğimiz çakışma/birleştirememe hatalarının çoğalmasını görebiliyoruz.

MacOS işletim sisteminin terminali Windows terminaline göre daha kapsamlı bir çalışma alanına sahip. Gerek Git kullanırken, gerek dosya ararken, gerek dosya açarken, gerek dosya silerken bizim başvurduğumuz bir yol. kullanıcılarını terminalleriyle bu kadar içli dışlı bir hale sokan işletim sisteminde Git gibi öğrenmesi zor ama öğrenince işlerimizi hızlı hızlı halledebilecğeimiz bir Sürüm Kontrol Sistemi ayrılmaz bir ikili oluyorlar.

Vakit Ayırdığınız için Teşekkürler

İletişim:
alpaltnceng@gmail.com
210601068
github.com/alpaltan
linkedin.com/in/alp-altan/