ÖMER KARAKEÇE 220601001

İzmir Bakırçay Üniversitesi Bilgisayar Mühendisliği 2. Sınıf

Linkedin : https://www.linkedin.com/in/%C3%B6mer-karake%C3%A7e-5774011a9/

Github : https://github.com/omerkrkcc

Medium : https://medium.com/@omerrkrkc

Versiyon Kontrol Sistemleri

Versiyon Kontrol Sistemleri Nedir ? Neden İhtiyaç Duyarız ?

Yazılım dinamik bir süreç gerektiren bir yapıdır . Yazılım geliştirme yapısının kurulması ve zamanla gelişmesi sebebiyle güncel teknolojilere ihtiyaç duyulmuştu.Böylelikle konfigürasyon yönetimine yönelik atılan ilk adım olan versiyon kontrol sistemleri ortaya çıkmıştır. Versiyon kontrol sistemleri yazılımcının süreç boyunca kaynak kod hakkında yaptığı değişikliklikleri takip etmelerine , yönetmelerine yardımcı olur . Projenin farklı versiyonlarını saklar ve yapılan her değişikliği geçmişine kaydeder. Versiyon kontrol sistemleri günümüzde, yazılımcılar ve yazılım geliştiren şirketler için önemli bir konuma gelmiştir. 1980'li yıllarda yazılım geliştirme süreçlerinin hız kazanması ile birlikte yazılımcıların ve proje üzerine çalışanların işleri yoğunlaşmaya başladı. Bunun üzerine yazılım geliştirme işlerinin kolaylaştırılması amacıyla ilk yerel (local) tabanlı versiyon kontrol sistemi olan RCS (Revision Contorl System) Walter F. Tichy tarafından açık kaynak kodlu olarak kullanıma açıldı. Böylelikle RCS'nin kaynak kodu, GNU Projesi'nin bir parçası olarak erişilebilir duruma geldi. RCS ile birlikte yazılım versiyon kontrol sistemleri üzerine ilgi ve çalışmalar artmıştır. Versiyon kontrol sistemleri ile yazılımların niteliği, erişilebilirliği artarken, projeye hız katması yönüyle zaman yönetiminden de kazanç elde edilmiştir.

Konfigürasyon yönetimine yönelik atılan ilk adım olan versiyon kontrol sistemlerini hakkında temel öngörüyü elde ettiğimize göre, versiyon kontrol sistemleri hakkında detaylara inebiliriz.

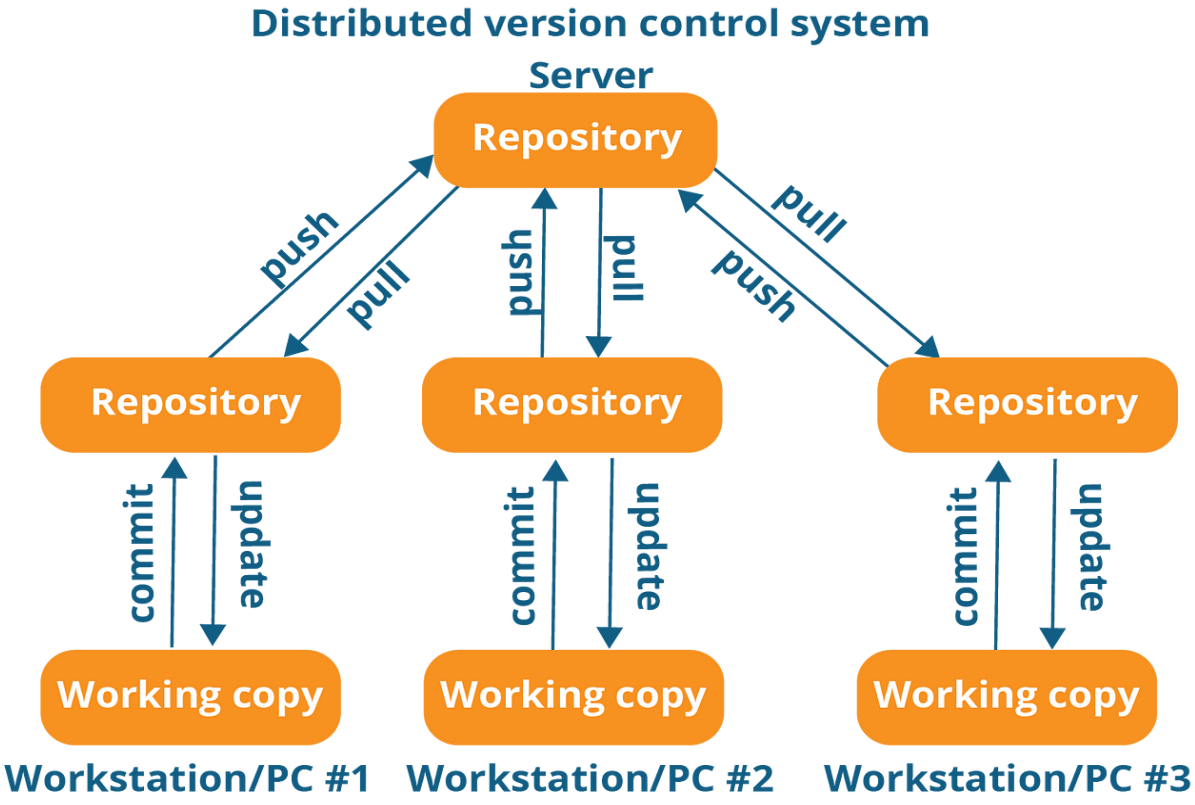
Yazılım projesi bir veya birden fazla grup geliştirici tarafından geliştirildiği için projede yapılan değişikliliklerin takibi ve yönetimi çok önemlidir . Versiyon kontrol sistemleri ile kaynak kod içerisinde kimin hangi değişikliği yaptığını, nerelerde değişiklik yaptığını, yapılan değişikliğin tarihini saatini görebiliriz. Versiyon kontrol sistemleri kodda yapılan her bir değişikliği bir veritabanında tutar. Bir hata durumunda yapılan geliştirmeler geri alınabilir veya kod üzerinde çalışılan önceki sürümlere ulaşılarak hatanın çözülmesi ve yönetilmesi kolaylaştırılır . Bu fayda ekip üyelerinin hızlı ve efektif bir ürün ortaya koymasına fayda sağlar . Sürüm kontrol sistemleri kaynak kodu üründen kaynaklanan herhangi ortaya çıkabilecek bir olumsuzluktan , insan hatalarından ve istenmeyen sonuçlardan korur. Versiyon kontrol sistemleri günümüzde

eşzamanlı çalışmaların çakışmalarının önlenmesine yardımcı olur. Yukarıda açıkladığım avantajlardan dolayı versiyon kontrol sistemleri günümüzde modern yazılım geliştirme ekiplerinin önemli bir parçası haline gelmiştir. Günümüzde popüler olarak kullanılan versiyon kontrol sistemlerine git, subversion, mercurial, bitbucket örnek verilebilir.

Versiyon Kontrol Sistemlerinin Kullanımının Avantajları ve Zorlukları

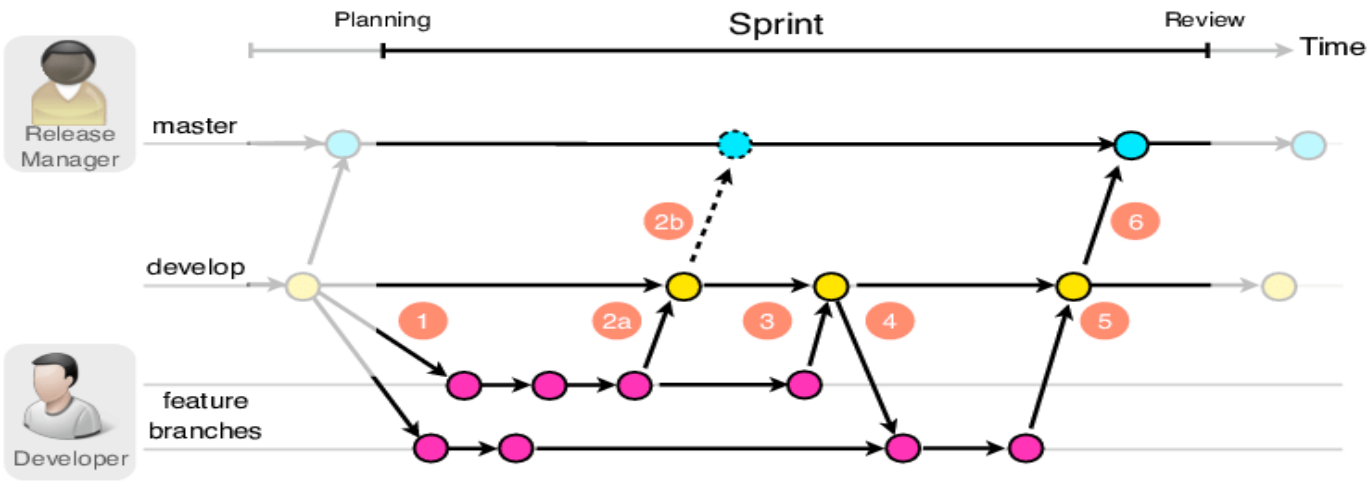
Versiyon kontrol sistemlerinin avantajlarını 5 temel aşamadan oluşur.

- **Uyumlu Ekip Çalışması :** Ekip üyeleri versiyon kontrol sistemi kullanarak yapmak istenilen dokümantasyona güvenli bir şekilde ulaşabilir ve istediği değişiklikleri yapabilir. Yapılan değişiklikler geriye alınabilir. Değişiklikler tamamlanınca doğru bir şekilde birleştirme (merge) işlemi yapılabilir. Versiyon kontrol sistemi sayesinde geleneksel yöntem olan herkese açık bir klasör kullanmak zorunda kalmazsınız . Geleneksel yöntem olan herkese açık klasör durumunda 2 kisinin aynı anda çalışması pek mümkün olamaz. Genelde tek kişi düzenlemeyi yaparken öbür kişi sadece okunabilir (readonly) kısmında olur . Eğer her 2 kişi de bu yapı üzerinde çalışmak isterse değişiklilik kayıt etme kısmında hata alırlar. Versiyon kontrol sistemi ile bir projede birden fazla kişinin çalışması mümkün olur. Her geliştirici değişikliklerini ayrı ayrı yapabilir. Kısacası tek bir proje üzerinde, birden çok kişi, birbirinden bağımsız bir şekilde çalışabilir. Böylelikle ekip içerisinde uyumluluk ve eşzamanlı çalışma mümkün olur.



• **Erişilebilirlik :** Proje süreci boyunca yapılan her değişikliliği izler ve birden çok versiyon oluşabileceğinden dolayı projeyi belli zaman dilimlerinde kayıt altına alır . Çalışma süreciniz boyunca en güncel versiyon üzerinde çalışırsınız daha önceki versiyonlar versiyon kontrol sisteminde depolanır. Herhangi bir geri dönülemez sorunla karşılaşılır ise eski sürüme dönülebilir ve hatalı değişiklilikler önlenebilir. Versiyonlar arasındaki farklılıkları inceleyebilirsiniz.

• **Paralel Geliştirme :** Projede var olan farklı özellikler ve değişiklilikler üzerinde çalışan farklı ekiplerin aynı anda çalışabilmesini sağlar . Yapıda bulunan her bir ekip kendi branchlariyla çalışıp en sonunda bu branchlar test edilip birleştirilerek yeni bir sürüm ortaya çıkarılır. Bu kısım proje geliştirme sürecine katkı sağlayan en önemli aşamalardan birisidir.



• **Açıklık ve Okunabilirlik :** Versiyon kontrol sistemlerinde değişiklikler tamamlandıktan sonra commit etmek istediğinde açıklama girilmesini isterler . Bu da projenizde sürümlere göre nerelerde ne değişiklik yaptığınızı, bu sürümü ne zaman yaptığınızı, ortaya çıkan sürümün yeni bir sürüm olmasını sağlayan niteliğide öğrenmiş olursunuz. Bu aşama hakkında örnek vermek gerekirse CVS versiyon kontrol sisteminin kullanımında commit işlemi yapılırken açıklama girilmesi zorunlu iken Git ve Bitbucket tarafında açıklama girilmesi önerilir.

• **Hata Tespiti :** Versiyon kontrol sistemleri ile bir yerde bulunan hatanın nerede olduğu daha iyi saptanabilir. Hatanın hangi sürümlerde ya da değişikliliklerden dolayı olduğunu ortaya çıkarmakta daha iyi bir ön izlenim sunar.

• **Şeffaflık :** Ortaya çıkan sürümler boyunca yapılan değişiklikleri şeffaf bir şekilde kullanıcıya sunar. Sürümlerde yaptığınız değişiklikleri inceleyebir ve karşılaştırabilirsiniz. Önceki sürüme erişebilirsiniz. Sürümünüzü önceki sürüme geri alabilir. Değişiklik yapılan döküman ile ilgili değişikliğin kimin yaptığını, hangi tarih ve saatte yaptığını inceleyebilirsiniz.

Versiyon Kontrol Sistemi Araçları

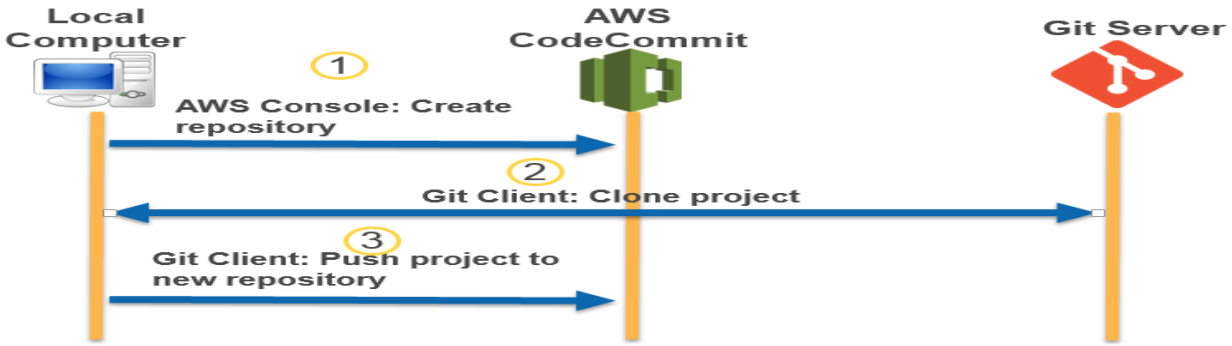
Günümüzde bir çok popüler versiyon kontrol sistemi araçları vardır. Burada en popüler 5 versiyon kontrol sistemi aracını inceleyeceğiz. Bunlar Git, Github, CVS ve AWS CodeCommit olacaktır.

Git : Git dağıtık bir versiyon kontrol sistemi aracıdır. Git Linux Torvalds ve beraberinde oluşan Linux çekirdeğini kodlayan bir ekip tarafından iş akışlarını yönetmek ve Linux kaynak kodunu versiyon kontrolü altında tutmak amacı ile geliştirilmiştir. Linux Torvalds ve ekibinin aslında git sürecini başlatma sebebi BitKeeper versiyon kontrol sisteminin lisanslarını iptal etmesiydi. Git kendisinin hızlı, kullanımı kolay, iş akışına uygun, tamamen dağıtık, büyük projeleri destekleyebilecek şekilde tasarlandığını ifade etmiştir. Git için optimizasyon önemlidir. Git, Github, GitLab, Bitbucket gibi popüler bulut hizmetleriyle entegre çalışabilir. Güçlü ve esnek olmasının yanı sıra iş akışına (branching) uygun olması sebebiyle günümüzde en popüler versiyon kontrol sistemi araçlarından birisi olmuştur.

GitHub : Github Tom Preston – Werner ve ekibi tarafından geliştirilen kısmen kapalı kaynaklı bir platformdur. Github'ın yapısındaki çoğu proje açık kaynaklıdır . Bir web tabanlı platformdur. Aynı zamanda GitHub Git'in deposu olarak tanımlanır. Kullanımı bakımından Git'e göre çok daha basittir. Arayüzü yazılım dünyasına yeni başlayan kişiler için kullanıışlıdır. GitHub'ın en büyük avantajlarından biri topluluk ekosisteminin güçlü olması olarak gösterilir

CVS : CVS açık kaynaklı bir merkezi versiyon kontrol sistemi aracıdır. Günümüzde dağıtık olmaması sebebiyle kullanımı oldukça azalmıştır. Dağıtık versiyon kontrol sistemleri merkezi sistemlere göre daha hızlı, daha esnek, daha güçlüdür. . Performans olarak rakiplerine göre yetersiz kalmaktadır. Kaynakların tek bir merkez üzerinden değerlendirilmesi güvenlik açısından negatif etkiler yaratır. CVS GNU Genel Kamu Lisansı (GPL) altında dağıtılmaktadır.

AWS CodeCommit : Amazon Web Servisleri (AWS) tarafından piyasa sürülen git tabanlı depolama hizmeti sunan bir platformdur. Güvenlik açısından versiyon kontrol sistemleri arasında en güvenilirlerinden biri olarak değerlendirilir. Platform Git'I temel alır. AWS CodeCommit kendisini güvenlikli, yüksek düzeyde ölçüklendirilebilir, tümüyle yönetilen bir kaynak yönetim hizmeti olarak tanımlamıştır. Günümüzde AWS hizmetlerinin kullanımı hızla artmaktadır.



Git Ve AWS CodeCommit Kullanıımı

Git : Kurulumunuzu git-scm.com/downloads adresinden yapınız. Kurulumunuzu yaptıktan sonra doğru olup olmadığını test etmek için bilgisayarınızın başlat kısmından Git Bash'I çalıştırınız. Git Bash kullanıcıların bir komut satırı ortamında çalışmasına olanak verir .Git Bask ayrıca Git GUI gibi arayüzlerle de entegre çalışabilir. Git Bash çalıştıktan sonra komut satırına git –version yazınız. Kullanım süreci boyunca herhangi bir hatayla karşılaşmamak için sürümün en güncel sürüm olduğundan emin olunuz.

Git Komutları

Burada Git komutlarımızı Git Bash üzerinden yazacağız.

• **git config :** Bu komut e posta ve kullanıcı adımızı yapılandırmaya yarar . Yapacağınız tüm projelerde bu kullanıcı adı ve e-posta adresini kullanırsınız.

Kullanıcı adı için → git config –global user.name "Omer Karakece"

E-posta için → git config –global user.email omerrkrkc@gmail.com

Kullanıcı adınızı ve e-postanızı tanımladıktan sonra aynı komutları girip yaptığınız yapılandırmanın doğru olup olmadığını kontrol edebilirsiniz. Eğer doğru komut ile yazdıysanız ilk yazdığınız komuttaki kullanıcı adını ya da emailinizi gösterir.

• **git init :** Bu komut git projesi oluşturmaya yarar . Kısacası mevcut olan konum boş bir repository haline getirir. Git init yazarak biz bu repository oluşturuyoruz. Fakat orada oluştuğundan nasıl emin olabiliriz ? Bunu anlamak için bulunan dizinde ls – a komutunu girmelisiniz. Böylelikle .git şeklinde gizli projeyi size gösterir.

• **git add :** Bu komut ile bulunulan dizinde istenilen döküman ya da tüm dökümanlar staging area adı verilen geçiş bölgesine gönderilir. Bu işlemden sonra git commit kullanılmalıdır .

• **git commit :** Bu komut ile staging area bulunan döküman ya da dökümanlar alınır ve git deposuna eklenir . Bu işlem için staging areaya ulaştıktan sonra git commit -m "ilk commit" diyerek projemizin mevcut bir kopyası oluşturulur. Depomuzda bulunan bütün versiyonları listelemek için ise **git log** komutunu kullanırız.

• **git status :** Bu komut ile git projemizde yapılan değişikliklikleri görebiliriz .Çalışma dizi ile git deponuz arasında fark yok ise git status commit olmadığını gösterir.

• **git diff :** Bu komut çalışan dizinimiz ile birlikte git depomuzda meydana gelen değişiklikleri gözlemliyoruz. Bir dökümanımızın içini değiştirdiğimizde git diff komutu ile yapılan değişikliğin olduğu dökümantasyonu ve ne değişikliğin yapıldığını gösterir. Bu değişikliği artık göstermemek için çalışan dizinimiz ile git depomuz uyumlu olmalı oncelikle git add ile geçiş bölgesine gönderilir. Ardından commit yapılarak git depomuza ekleriz. Bu işlemler yapıldıktan sonra git log dendiğinde yayımladığımız sürümleri gözlemliyoruz. Ayrıca git status yazarsanız önceki yapılan

değişikliklere karşı verilen modified uyarsı yerine artık commit yok uyarısı gösterilecek. Yani her koşulda mevcut calısan dizinimiz ile git depomuzda fark kalmamış oluyor. Bunu test etmek için git diff yazıp hiç bir değişiklik olmadığını gözlemleyebilirsiniz. Ayrıca git diff –staged ile geçiş bölgesi ile git depomuz arasındaki farklılıkları gözlemleyebiliriz.

git checkout : Git checkout ile dökümantasyonumuzda yaptığımız değişiklikleri geri alabiliriz . Mesela gittiniz bir dökümantasyondan bir şeyleri sildiniz ya da gittiginiz dökümantasyonun kendisini sildiniz. Dökümantasyonun adı omer olsun. Onu git checkout – omer yazarak kurtarabilirsiniz. Bu komut ile ayrıca deponuzda oluşturduğunuz sürümler arası geçişler yapabilirsiniz. Örneğin git checkout 53e546e45e5d5a5 -- . (ilgili versiyonun hash kodu) yaparak ilgili sürüme geçiş yapabilirsiniz.

Git ile commitleme – Branch oluşturma – Remote repository bölüm 2 de pdf üzerinden detaylıca anlatılmıştır.

AWS CodeCommit : AWS CodeCommit için herhangi bir şey indirmenize gerek kalmadan tüm ihtiyacınızı https://aws.amazon.com/tr/codecommit/ üzerinden gerçekleştirebilirsiniz. Kullanım acısından oldukca basittir. Yeni başlayanlara uygundur . Git gibi bash tabanlı konsollar kullanmanıza gerek kalmadan tek bir web panelinden pushlama, repo olusturma, commit etme, branch yönetimi, değişiklikliklerin geçmişi ve bünyesinde barındırdıgı AWS CodePipeline ile sürekli entegrasyon ve dağıtım (CI / CD) süreçlerinizi düzenleyebilirsiniz. AWS CodeCommit'in karmaşık olmayışı tek bir panel üzerinden sadece gerekli butonlara tıklanarak işlevlerinin yapılabileceğinden dolayı detaya inmek istemedim. AWS CodeCommit hakkında daha fazla bilgi almak istiyenler https://aws.amazon.com/tr/codecommit/ sitesini ziyaret edebilir. Türkçe dil desteklerinden faydalanarak AWS teknolojilerini öğrenme yolculuğuna başlayabilirler.

KAYNAKÇA :

- **https://git-scm.com/doc**
- **https://www.youtube.com/watch?v=tRZGeaHPoaw**
- **https://www.gitbook.com/?utm_source=legacy&utm_medium=redirect&utm_campaign=close_legacy**
- **https://dev.to/koraybarkin/git-de-branch-nedir-nasil-olusturulur-em6**
- **https://www.atlassian.com/git/tutorials/what-is-version-control**
- **https://www.foch.com.tr/versiyon-kontrol-git-svn-cvs.html**
- **Doç. Öğr. Üyesi Zekeriya Anıl GÜVEN, İzmir Bakırçay Üniversitesi Yazılım Mühendisliği Temelleri Dersi 10. ve 11. Hafta Sunumları**