16.05.2023

İçindekiler :

* versiyon kontrol sistemleri tanıtılmış ve avantaj ve dezavantajları açıklanmıştır
* versiyon kontrol sistemlerinden Git ve Mercurial özellikleri açıklanmıştır Windows ortamında kullanımı açıklanmıştır
* Github ve Medium adresleri :
* [*https://github.com/MehmetsirinAKIN/Versiyon\_kontrol\_sistemleri*](https://github.com/MehmetsirinAKIN/Versiyon_kontrol_sistemleri)
* <https://medium.com/@msrnakin4949>



**Mehmet ŞirinAkın 210601015 BİLGİSAYAR MÜHENDİSLİĞİ YAZILIM MÜHENDİSLİĞİ TEMELLERİ (BIL 102)ÖDEV #2**

1. **VKS'lerinin Tanıtımı:** Versiyon kontrol sistemleri, yazılım geliştirme sürecindeki dosyaların ve değişikliklerin izlenmesini, yönetilmesini ve takip edilmesini sağlayan yazılımlardır. VKS'lerinin temel amacı, bir projenin farklı sürümlerinin yönetimini kolaylaştırmak ve bir ekip tarafından ortak çalışılan bir kod tabanı oluşturulmasını sağlamaktır. VKS'leri, geliştiricilerin kodu paylaşmasını, değişiklikleri takip etmesini, geri almasını ve birleştirmesini sağlar.



1. **Kullanımın Avantajları:**

**a) Paralel Geliştirme**: VKS'leri, birden fazla geliştiricinin aynı anda çalışmasına olanak tanır. Her geliştirici, kendi kopyasında değişiklikler yapabilir ve VKS, bu değişiklikleri birleştirerek sorunsuz bir şekilde senkronize eder.

**b) Geri Alabilme**: VKS'leri, hatalı bir kod değişikliği yapıldığında veya yanlış bir yönde ilerlendiğinde geri almayı sağlar. Önceki sürümlere kolaylıkla dönerek hataları düzeltebilir ve sağlam bir kod tabanı oluşturabilirsiniz.

**c) İzlenebilirlik:** VKS'leri, dosyaların ve değişikliklerin tam bir tarihçesini sunar. Hangi geliştiricinin ne zaman hangi değişiklikleri yaptığı, hangi hataların düzeltildiği gibi bilgilere kolayca ulaşabilirsiniz.

**d) Ekip İşbirliği:** VKS'leri, ekipler arasındaki işbirliğini kolaylaştırır. Herkesin aynı kaynak kodu üzerinde çalışmasına olanak tanır, çakışan değişiklikleri yönetir ve daha verimli bir şekilde birlikte çalışmayı sağlar.

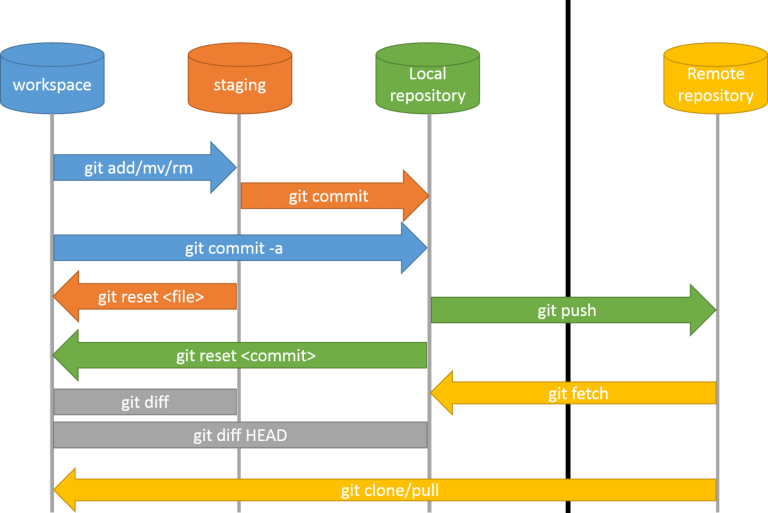
1. **Kullanımın Zorlukları:**

**a) Öğrenme Eğrisi:** VKS'lerin kullanımı başlangıçta biraz karmaşık olabilir. Yeni bir VKS aracı öğrenmek ve kullanmaya alışmak zaman alabilir. Farklı komutlar, dalgalanmalar ve birleştirme süreçleri gibi bazı karmaşıklıklarla karşılaşabilirsiniz. Ancak, zamanla VKS'lerin işleyişini ve en iyi uygulamalarını anladıkça, bu zorlukların üstesinden gelebilirsiniz.

**b) Çakışan Değişiklikler:** Birden fazla geliştiricinin aynı dosyada aynı anda değişiklik yapma riski vardır. Bu durumda, VKS'lerde çakışmalar meydana gelebilir. Çakışan değişiklikleri yönetmek ve birleştirmek zaman alıcı olabilir. Ancak, VKS'ler genellikle bu tür çakışmaları yönetmek için araçlar ve stratejiler sunar, böylece çakışmaların en aza indirilmesi ve sorunsuz bir şekilde çözülmesi sağlanır.

**c) Depolama ve Yedekleme:** VKS'lerde büyük miktarda kod ve dosya depolama gereksinimi olabilir. Proje geçmişinin tam bir kopyası tutulduğu için depolama alanı gereksinimleri artabilir. Ayrıca, VKS'nin sağladığı yedekleme özelliklerini düzenli olarak kullanmak önemlidir, böylece verilerin kaybolma riski minimize edilir.

1. **VKS Aracı: Git,** en popüler ve yaygın kullanılan VKS araçlarından biridir. Hem Windows hem de MacOS üzerinde kullanılabilir ve açık kaynaklı bir dağıtık VKS sistemidir. Git'in özellikleri ve kullanımı aşağıda açıklanmıştır:

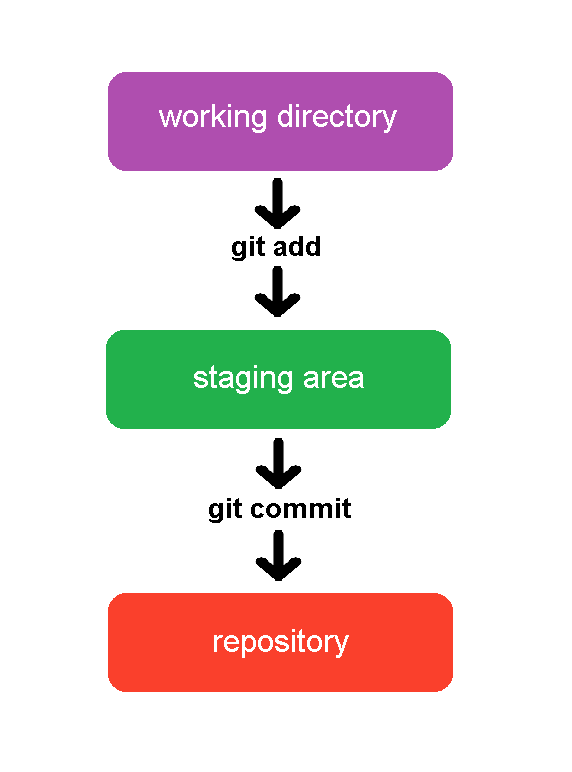


**a) Özellikler:**

* Dağıtık yapı: Git, her geliştiricinin kendi kopyasında çalışmasına olanak tanır ve değişikliklerin merkezi bir sunucu üzerinde birleştirilmesi gerekmez.
* Hızlı ve Verimli: Git, hızlı ve verimli bir şekilde büyük projeleri yönetebilir. Yüksek performanslı bir VKS aracı olarak bilinir.
* Dal (Branch) Desteği: Git, projenin farklı özelliklerini veya hatalarını izole etmek için dallar oluşturmayı sağlar. Her dal, ayrı bir çalışma alanı sağlar ve değişikliklerin sorunsuz bir şekilde birleştirilmesini sağlar.

**b) Kullanımı:**

* Git'in yüklenmesi: Git'i Windows veya MacOS üzerinde kullanmak için öncelikle resmi web sitesinden indirmeniz ve kurmanız gerekmektedir. Kurulum süreci genellikle kullanıcı dostu bir arayüzere tasarlanmıştır ve adım adım yönergeler sunar.
* Bir Git Depo oluşturma: Git projenizi yönetmek için bir Git deposu (repository) oluşturmanız gerekmektedir. Bu depo, proje dosyalarınızın ve değişikliklerinizin merkezi bir noktasıdır. Depoyu oluşturmak için **git init** komutunu kullanabilirsiniz.
* Dosyaları İzleme: Oluşturduğunuz Git deposunda takip etmek istediğiniz dosyaları belirlemeniz gerekmektedir. Bu dosyaları izlemek için **git add** komutunu kullanabilirsiniz. Örneğin, **git add dosya1.txt** komutuyla **dosya1.txt** dosyasını izleme altına alabilirsiniz.
* Değişiklikleri Kaydetme: Yaptığınız değişiklikleri kaydetmek için **git commit** komutunu kullanabilirsiniz. Bu komut, değişiklikleri birer paket halinde kaydederek depoya ekler. **git commit -m "Değişiklik açıklaması"** komutuyla birlikte değişikliklerinizi açıklayıcı bir mesajla kaydedebilirsiniz.
* Dal Yaratma ve Birleştirme**:** Git, dalları oluşturmanıza ve değişiklikleri bu dallar arasında birleştirmenize olanak tanır. Bir dal oluşturmak için **git branch** komutunu kullanabilirsiniz. Bir dalı diğerine birleştirmek için ise **git merge** komutunu kullanabilirsiniz.
* Uzak Depolarla İşbirliği**:** Git, uzak depolarla çalışmayı da destekler. Başka bir sunucu veya hizmet üzerindeki bir Git deposuyla işbirliği yapmak için **git remote** komutunu kullanabilirsiniz. Bu komut, uzak depoyu projenize eklemenizi ve değişiklikleri paylaşmanızı sağlar.



1. **VKS Aracı: Mercurial**, bir diğer popüler VKS aracıdır ve hem Windows hem de MacOS üzerinde kullanılabilir. Mercurial'in özellikleri ve kullanımı aşağıda açıklanmıştır:



**a) Özellikler:**

* Basit ve Kullanıcı Dostu Arayüz: Mercurial, kullanıcı dostu bir arayüz sunar ve yeni başlayanlar için kolayca erişilebilir bir VKS aracıdır.
* Hızlı ve Verimli İşlemler: Mercurial, büyük projelerde bile hızlı ve verimli bir şekilde çalışabilir. Yüksek performanslı ve ölçeklenebilir bir VKS aracı olarak bilinir.
* Kolay Yönetim:Mercurial, değişiklikleri yönetmek ve birleştirmek için kolay anlaşılabilir komutlar ve işlem adımları sunar.

**b) Kullanımı:**

* Mercurial'in Yüklenmesi: Mercurial'i Windows veya MacOS üzerinde kullanmak için resmi web sitesinden indirmeniz ve kurmanız gerekmektedir. Kurulum süreci genellikle kullanıcı dostu bir arayüz üzerinden gerçekleştirilir.
* Bir Mercurial Depo Oluşturma: Mercurial ile projenizi yönetmek için bir Mercurial depo oluşturmanız gerekmektedir. Depo, proje dosyalarınızın ve değişikliklerinizin merkezi bir noktasıdır. Depoyu oluşturmak için **hg init** komutunu kullanabilirsiniz.
* Dosyaları İzleme: Oluşturduğunuz Mercurial deposunda takip etmek istediğiniz dosyaları belirlemeniz gerekmektedir. Dosyaları izlemek için **hg add** komutunu kullanabilirsiniz. Örneğin, **hg add dosya1.txt** komutuyla **dosya1.txt** dosyasını izleme altına alabilirsiniz.
* Değişiklikleri Kaydetme: Yaptığınız değişiklikleri kaydetmek için **hg commit** komutunu kullanabilirsiniz. Bu komut, değişiklikleri birer paket halinde kaydederek depoya ekler. **hg commit -m "Değişiklik açıklaması"** komutuyla birlikte değişikliklerinizi açıklayıcı bir mesajla kaydedebilirsiniz.
* Dal Oluşturma ve Birleştirme: Mercurial, dalları oluşturmanıza ve değişiklikleri bu dallar arasında birleştirmenize olanak tanır. Bir dal oluşturmak için **hg branch** komutunu kullanabilirsiniz. Bir dalı diğerine birleştirmek için ise **hg merge** komutunu kullanabilirsiniz.
* Uzak Depolarla İşbirliği: Mercurial, uzak depolarla çalışmanızı da destekler. Başka bir sunucu veya hizmet üzerindeki bir Mercurial deposuyla işbirliği yapmak için **hg clone** komutunu kullanabilirsiniz. Bu komut, uzak depoyu projenize eklemenizi ve değişiklikleri paylaşmanızı sağlar.

1. **Sonuç:** Versiyon kontrol sistemleri (VKS), yazılım geliştirme sürecinde büyük önem taşır. Bu makalede VKS'lerinin tanımını yaptık ve kullanımının avantajları ve zorluklarına değindik. Ayrıca, Windows veya MacOS üzerinde kullanılan iki farklı VKS aracı olan Git ve Mercurial'i inceledik. Her iki araç da geliştirme sürecini kolaylaştıran ve ekip işbirliğini destekleyen özelliklere sahiptir. Git, dağıtık yapı ve güçlü dal desteği ile öne çıkarken, Mercurial basit arayüzü ve kullanıcı dostu işlemleriyle tercih edilebilir. Her iki VKS aracı da yazılım geliştirme sürecinde verimlilik, izlenebilirlikve güvenilirlik sağlamak için önemli bir rol oynamaktadır.

VKS'ler, geliştiricilerin kodu paylaşmasını, değişiklikleri izlemesini, geri almasını ve birleştirmesini sağlayarak yazılım projelerinin daha düzenli ve organize bir şekilde yönetilmesini sağlar. Paralel geliştirme, geri alabilme, izlenebilirlik ve ekip işbirliği gibi avantajlarıyla VKS'ler, yazılım geliştirme sürecini daha etkili hale getirir.

Ancak VKS'lerin kullanımında bazı zorluklar da bulunmaktadır. Öğrenme eğrisi, çakışan değişikliklerin yönetimi ve depolama/yedekleme gereksinimleri gibi zorluklarla karşılaşılabilir. Bu zorlukların üstesinden gelmek için VKS'lerin doğru bir şekilde öğrenilmesi, iyi uygulamaların benimsenmesi ve düzenli yedekleme yapılması önemlidir.

Sonuç olarak, VKS'ler yazılım geliştirme sürecinde vazgeçilmez bir araçtır. Git ve Mercurial gibi VKS araçları, kullanıcılarına güçlü özellikler sunarak geliştirme sürecini yönetmeyi kolaylaştırır. Bu araçları etkili bir şekilde kullanarak, yazılım projelerinizde daha verimli, düzenli ve işbirlikçi bir çalışma ortamı oluşturabilirsiniz.