VERSİON CONTROL

HAZIRLAYANLAR;

GÜLZADE ODABAŞ

SELCAN KAZAN

İÇERİK;

-Versiyon Kontrol Sistemi Nedir ?

-Neden Versiyon Kontrolüne İhtiyaç Duyarız?

-Versiyon Kontrol Sistem Tipleri Nelerdir?

1-Local Versiyon Kontrol Sistemleri

2-Merkezi Versiyon Kontrol Sistemleri

*CVS (CONCURRENT VERSİONS SYSTEM)
*SVN (SUBVERSİON VERSİONS SYSTEM)

3-Dağıtık Versiyon Kontrol Sistemlerİ

*MERCURİAL NEDİR?

*BİTKEEPER NEDİR?

*GİT NEDİR?

*Neden Git Kullanmalıyız?

Versiyon Kontrol Sistemi Nedir ?

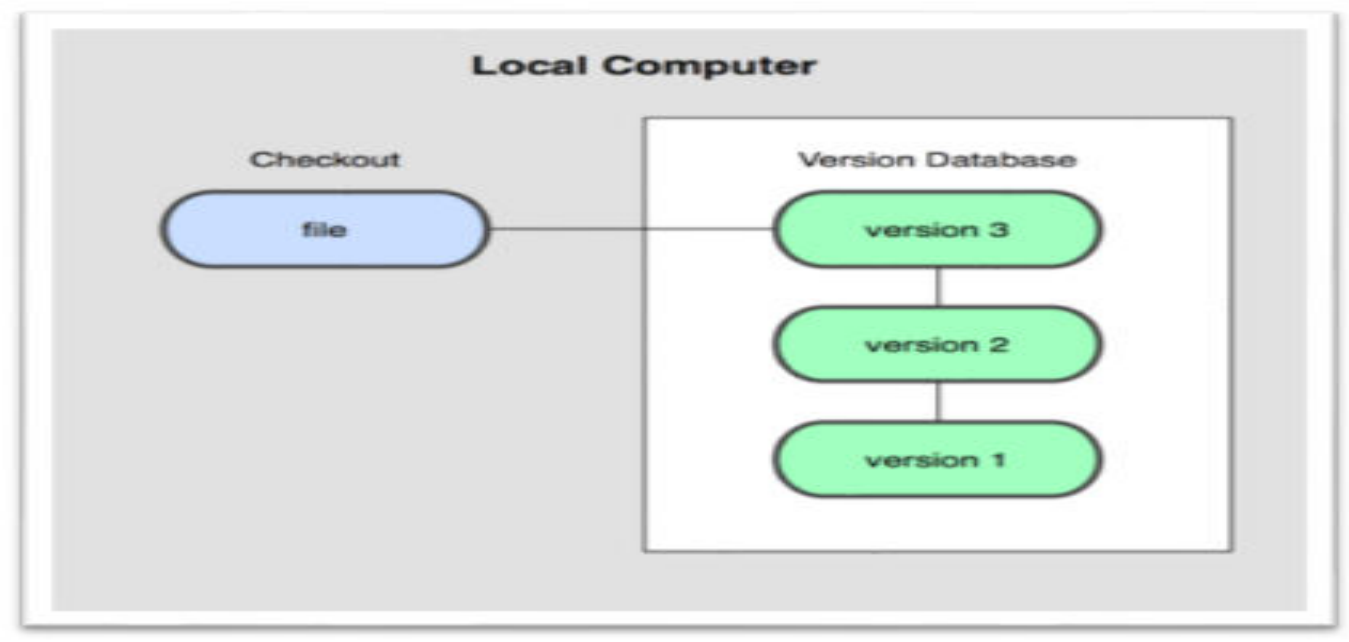
- Version Control System bir döküman (yazılım projesi, ofis belgesi…) üzerinde yaptığımız degişiklikleri adım adım kaydeden ve isterseniz bunu internet üzerinde depoda (respository) saklamamızı ve yönetmemizi sağlayan bir sistemdir. Git, SVN, BitKeeper, Mercurial birer sürüm kontrol sistemleridir.
- Kısaca, uygulamanızın kaynak kodunun yönetimini sağlayan uygulamalardır.

Neden Versiyon Kontrolüne İhtiyaç Duyarız?

- Projelerde birden fazla kişi çalıştığını düşünürsek gelişimin hızlanmasını sağlar.
- Projede geliştirme yaparken, bulunduğumuz konuma nereden geldiğimizi anlamak için eski ve yeni kodumuz arasında karşılaştırma yapmamızı sağlar.
- Projede hatayla karşılaştığımız durumlarda eski kod kaydına dönmemizi sağlar.
- Açık kaynaklı projeleri baz alınarak geliştireceğimiz yeni projelerde süreci kolaylaştırmayı sağlar.

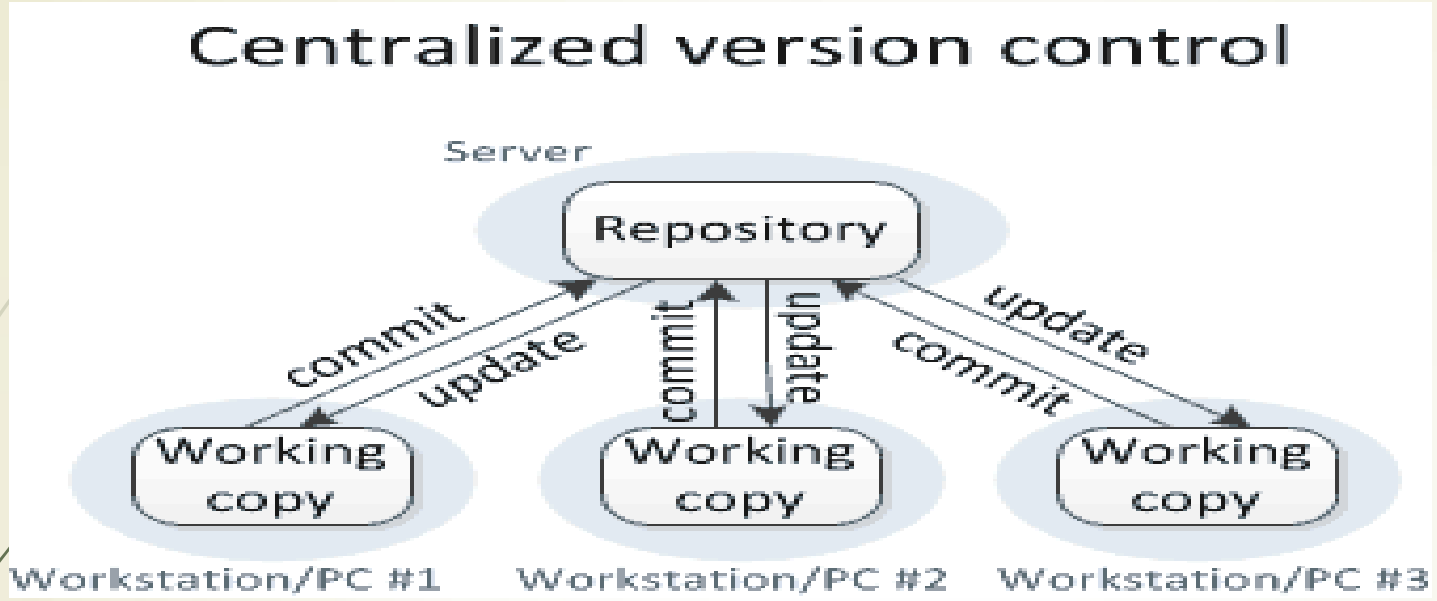
Versiyon Kontrol Sistem Tipleri Nelerdir?

1-Local Versiyon Kontrol Sistemleri



- En eski versiyon kontrol sistemi yaklaşımıdır. Çalıştığımız projemiz ve yaptığımız değişiklikler kullanıcı makinası üzerindeki veritabanında tutulur. Her yaptığımız commit bir versiyon olarak tutulur ve commit değerine hash ataması yapılarak her versiyon birbirinden ayırt edilir. Ayrıca versiyon görüntüleme imkanını sağlar. Ancak bu sistemde yanlızca bir kullanıcı etkin bir şekilde çalışabilir.

2-Merkezi Versiyon Kontrol Sistemleri



- **Birden fazla kişinin bir proje üzerinde etkin çalışması için ortaya atılmiş versiyonlama sistemidir. CVS, SVN birer merkezi versiyon kontrol sistemleridir. Bu sistemde proje ortak bir respositoryde tutulur ve birden fazla geliştirici aynı respository üzerinde checkout ve commit işlemlerini gerçekleştirir. Bu yöntemde herkesin projede katkı sağlamasının yanısıra bazı ciddi sorunları vardır. Tek merkezli sunucu 1 saatliğine arızalanması durumunda , kullanıcılar 1 saat boyunca çalışmalarını veya çalıştıkları projenin sürümlenmiş kopyalarını kaydetmeleri mümkün olmayacaktır.**

1. CVS (CONCURRENT VERSİONS SYSTEM)

- CVS , (Concurrent Versions System) açık kaynak olarak geliştirilmiş olan kaynak kod kontrol sistemidir.
- Kaynak yazıların takibini kolaylaştırır.
- Kaynakların tek merkezden yönetilmesine olanak tanır.
- Dosyaların istediğiniz sürümlerini çekmenize fırsat verir ve sizin yerinize bütün dosyalara sürüm numaraları verir.
- Programcı istediği tarihteki kaynak kodlara erişip üzerinde çalışma yapabilir.
- Programcı, diğer proje üyelerinin yazdığı kodları görebilir, onlara ilave yapabilir. Yaptığı bu ilaveleri diğer programcılar takip edebilir

2. SVN (SUBVERSİON VERSİONS SYSTEM)

- SVN açık kaynaklı (open source) versiyon kontrol sistemidir.SVN , CVS örnek alınarak yapılmıştır. Amaç CVS'de daha iyi bir versiyon kontrol sistemi oluşturmaktır. Bu yüzden SVN birçok CVS özelliğine sahiptir.
- Bazı özellikleri söyledir;

* SVN dizinlerin de normal dosyalar gibi versiyonlarını oluşturur.

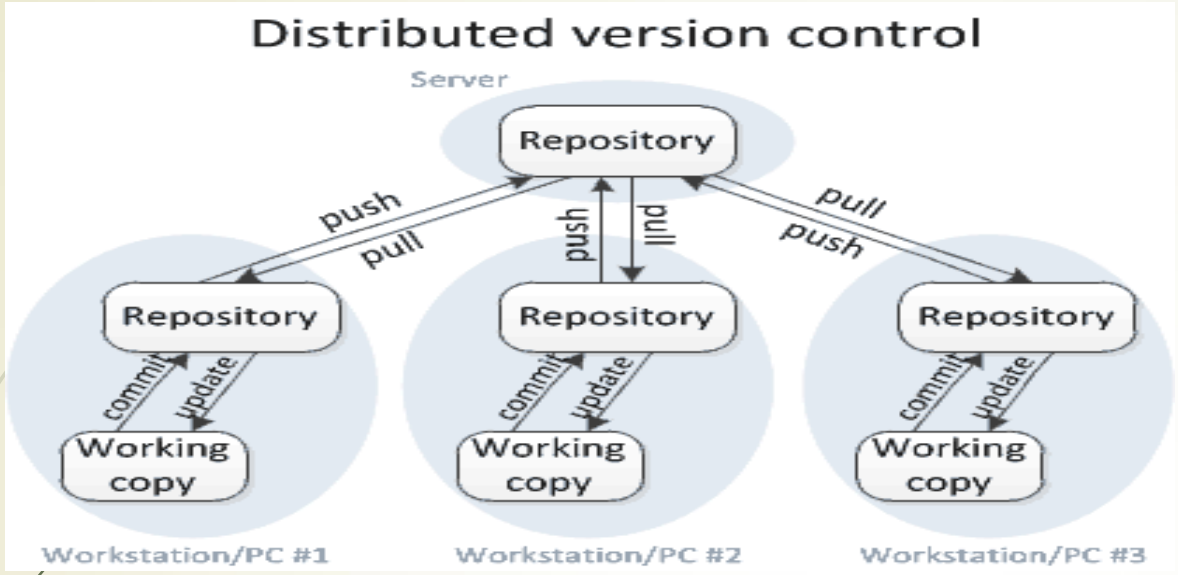
* Kopyalama, silme ve isim değistirme işlemlerinde Subversion tarafından yeni versiyonlar olusturulur.

* File locking (dosya kitleme) mekanizması kullanılarak, dosyaların üzerinde değişiklik yapılması engellenebilir.

* Subversion bir Apache webserver üzerinde erisilebilir hale getirilebilir

*Svnserve komutuyla Subversion, versiyon kontrol serveri olarak görev yapabilir.

3-Dağıtık Versiyon Kontrol Sistemler



Merkez versiyon sistemlerinin geliştiricilerin offline çalışabilmesi ve respository'nin zarar görmesi durumunda geri getirme gibi eksikliklerinden dolayı ortaya atılmış bir versiyon sistemidir. Git, Mercurial, BitKeeper.. Dağıtık versiyon sistemleri örnekleridir.
Bu sistemlerde merkezi bir respository yoktur. Proje üzerinde çalışan her makina projenin kopyasını tutmaktadır. Geliştiriciler proje üzerinde değişiklik yapmak veya proje geçmişine göz atmak istediklerinde uzak depo ile iletişime geçmek zorunda değildir. Sunuculardan biri çökerse, ve o sunucu üzerinde ortak çalışma yürüten sistemler varsa, geliştircilerden birinin projeyi sunucuya geri yükleyerek sistem kurtarılabilir.
Kısacası aynı projede farklı geliştiriciler farklı biçimlerde çalışma yürüterek, farklı iş akışları ile çalışabilmemizi sağlar

BİTKEEPER NEDİR?

- BitKeeper, çok büyük projelere küçücük ölçeklenen hızlı, orijinal dağıtılmış kaynak yönetim sistemidir
- Arayüzü kullanımı kolaydır.
- İkili dosyalar için kaynak depoları doldurmak yerine sunucu bulutu kullanır.
- Oluşturma, silme ve yeniden adlandırma gibi dosya işlemlerinin doğruluğunu kontrol eder.
- Dosya yazma işlemlerini ve hata düzeltmek için kullanılır.
- Kaynak anonslar anında kullanılabilir.
- Hızlıdır. Yüksek performans ve çok büyük depolar için ölçeklendirilebilr.

MERCURİAL NEDİR?

- Mercurial, yazılım geliştiricileri için bağımsız bir sürüm denetim aracıdır. Çoğunlukla Python kullanılarak yazılmıştır ancak C ile yazılan bazı yerleri de vardır. Önceleri sadece Linux için yazılmış olup daha sonra Windows ve Mac OSX sürümleri de çıkmıştır. Mercurial bir komut satırı programıdır ancak grafik arayüz eklentileri de vardır.

- Kısaca ücretsiz dağıtılmış kaynak kontrol yönetim aracıdır.Verimli bir şekilde her boyutta projeleri yönetir ve kolay sezgisel bi arayüze sahiptir.

GİT NEDİR?

- Git, kısa süre içerisinde yazılımcıların vazgeçilmezleri arasına giren bir sürüm/versiyon kontrol sistemidir. Yazdığımız projeleri, bilgisayarımızda ya da harici disklerde binbir tehlike altında değilde internet üzerinde tutmamızı ve yönetmemizi sağlayan bir sistemdir.

Neden Git Kullanmalıyız?

- **1- Versiyon yönetim kolaylığı**

Projenizi bilgisayarınızda, bir harici diskte ya da bir bulut sisteminde tutuyorsanız, projede bir hata yapma veya projenizin başına bir sorun gelmesi gibi olasılıkları göz önünde bulundurarak sürekli projenizi yedekleme ihtiyacı duyarsınız. Bu sebeple projenizi versiyon versiyon kaydetmeye başlarsınız.

Projenizin büyüklüğüne göre bu işlemi yapmak oldukça karmaşık halleralabilir. Eğer Git kullanırsanız böyle bir sorununuz olmaz. İstediğiniz zaman projenizin son kaydettiğiniz haline ulaşabilir veya herhangi bir günki haline dönüş yapabilirsiniz.

2- Birden fazla kişinin eş zamanlı aynı proje üzerinde çalışması;

Aynı proje üzerinde, birden fazla kişi ile eşzamanlı olarak ya da farklı zamanlarda çalışıyorsanız kodlarınızı birleştirmek ya da kod alışverişi yapmak oldukça çetrefillidir. Fakat git kullanıyorsanız bu işlemleri yapmanız hiç de zor olmayacaktır.

3- Offline kullanılabilmesi

Bu özellik diğer versiyon kontrol sistemlerinden farklı olarak hiçbir ağa bağlı olmadan projenizi geliştirebilmenizi sağlar. Çoğu sistemde proje sürekli merkezi bir noktada dururken, Git'te her kullanıcının bilgisayarında projenin bir kopyası mevcut olur ve bu sayede offline olarak da proje geliştirebilirsiniz.

4- Hızlı olması ve az yer kaplaması

Ayrıca Git, bilgisayarınızda fazla yer kullanımını önler ve oldukça hızlı çalışır.

TEŞEKKÜRLER...