Versiyon Kontrol Sistemleri - Git ve Github ile Proje Yönetimi

Kenan Polat 12011037

https://github.com/kenanpoll

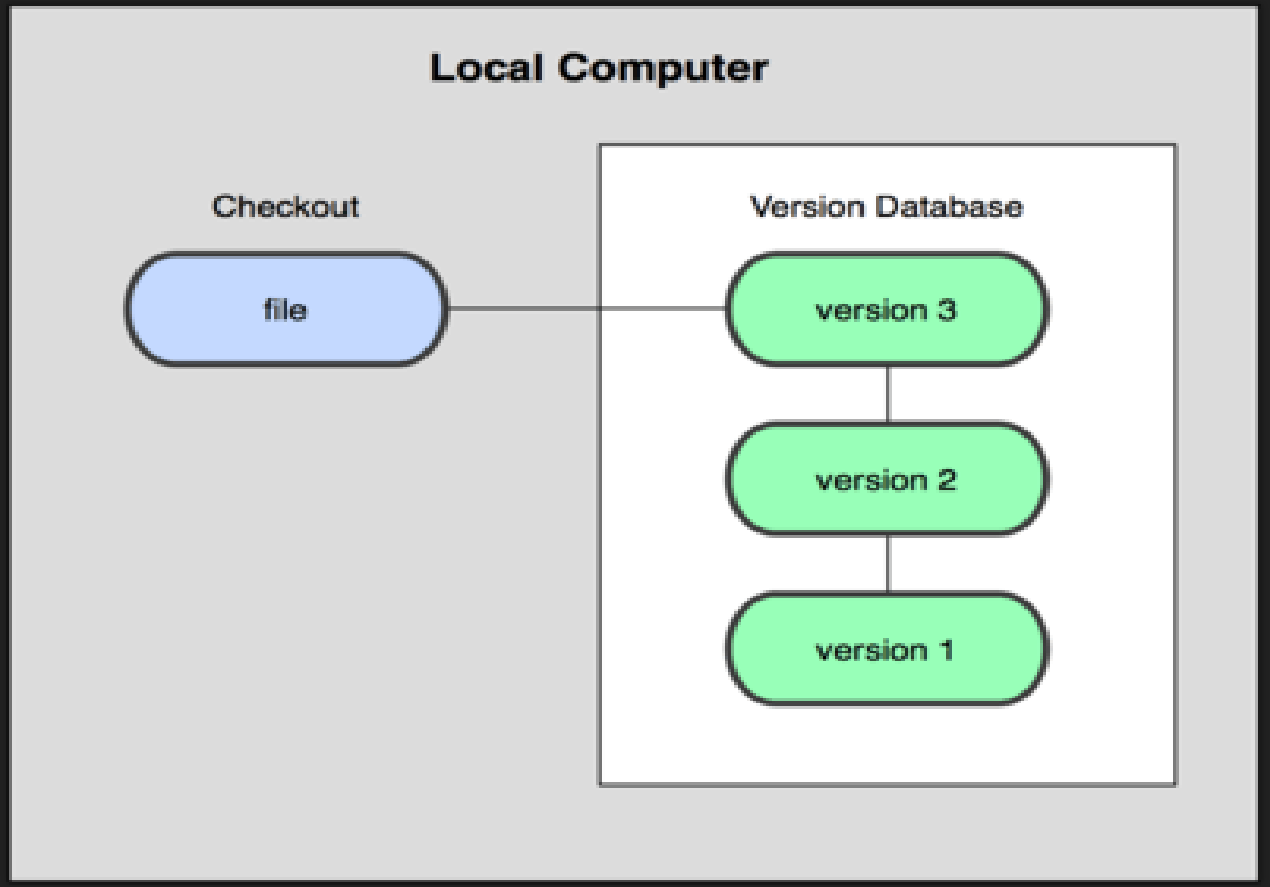
Versiyon Kontrol Sistemi Nedir?

- Versiyon kontrolü, yazılım geliştirmedeki anlamından bağımsız olarak, bir dokümanın üzerinde yapılan değişikliklerin yönetilmesidir. Burada bahsedilen "doküman", Bir ofis belgesi, web sitesi, hatta bir uygulama programı bile olabilir.
- Yazılım geliştirme üzerinde özelleştirirsek, versiyon kontrolü bir yazılımın revizyonlar halinde güncellenerek geliştirilmesini sağlar. Böylece geliştirilen yazılımın geçtiği süreçler rahatça takip edilebilir*

Neden Versiyon Kontrol Sistemi Kullanılır?

- Birden fazla kişinin beraber çalıştığı projelerde proje gelişiminin hızlanması için kullanılır.
- Açık kaynak projelerde hali hazırda tamamlanmış projeler baz alınarak geliştirilen yeni projelerde süreci kolaylaştırmak için kullanılır.

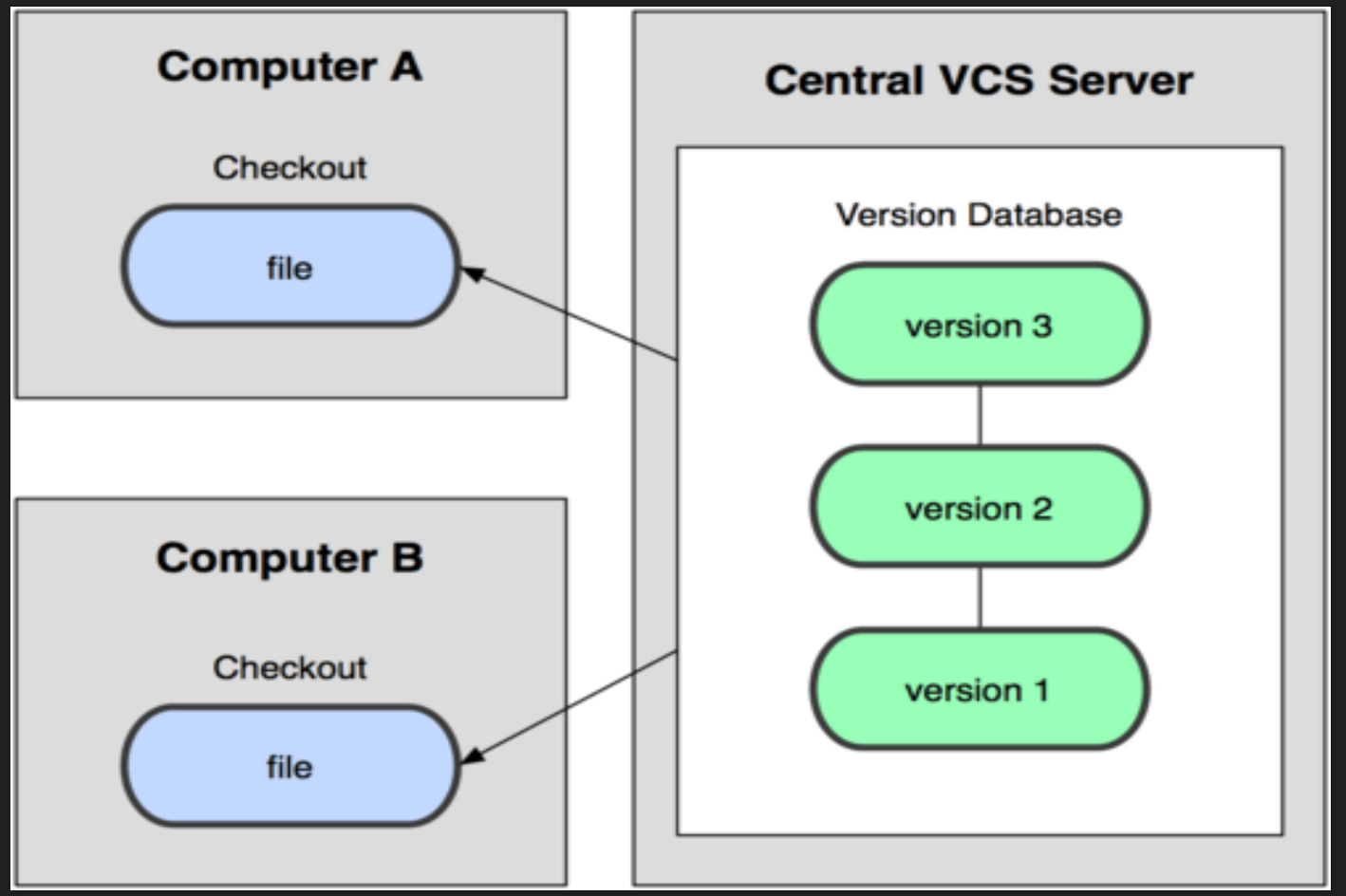
Yerel Sürüm Kontrol Sistemleri



Merkezi Versiyon Kontrol Sistemi

- Merkezi versiyon yönetim sistemlerinde tüm kullanıcılar bütün değişikliklerini direkt olarak uzaktaki depo (repository) üzerinde yaparlar.
- Son revizyonu yapan kullanıcının yeni bir değişiklik yapıp uzak sunucuya göndermesi büyük olasılıkla bir sorun yaratmaz.
- Başka bir kullanıcı kendi değişikliklerini uzak depoya gönderip nihai hale getirmek istediğinde (commit) uzak sunucudaki kodun stabilitesinin korunması için VKS ikinci kullanıcıya önce uzak depodaki kodu çalıştığı kodun üzerine indirmesini (update) isteyecek.

Merkezi Versiyon Kontrol Sistemi



Merkezi Versiyon Kontrol Sistemi

- Uzak sunucudan gelen dosyalar sizin geliştirdiğiniz kodlar ile uygun şekilde birleştirilecek. (merge)
- Ardından ikinci kullanıcı değişikliklerini sorunsuzca uzak depoya yollayabilecek.
- **Merge işlemi versiyonun yönetiminin en can alıcı noktasıdır. Diyelim ki iki kullanıcı da bir dosyada aynı satırlar üzerinde değişiklik yaptılar. Bu durumda birleştirilen dosyada hangi kullanıcının değişikliği saklanacak?**

Merkezi Versiyon Kontrol Sistemi

- İşte bu ikileme versiyon kontrolü sözlüğünde conflict (çatışma) denir
- Bir dosya üzerinde iki değişiklik çatışırsa çoğu zaman bu durumun düzeltilmesi için elle müdahele gerekir.
- Çatışmalar çözüldükten sonra gereken değişiklikler yapılarak yeni versiyon uzak sunucuya yollanabilir.

Merkezi Versiyon Kontrol Sistemi

- Ne var ki, bu yöntemin de ciddi bazı sıkıntıları vardır. En aşikar sıkıntı, merkezi sunucunun arızalanması durumunda ortaya çıkacak kırılma noktası problemidir.
- Sunucu bir saatliğine çökecek olsa, o bir saat boyunca kullanıcıların çalışmalarını sisteme aktarmaları ya da çalıştıkları şeylerin sürümlenmiş kopyalarını kaydetmeleri mümkün olmayacaktır.
- Merkezi veritabanının sabit diski bozulacak olsa, yedekleme de olması gerektiği gibi yapılmamışsa, elinizdeki her şeyi* kaybedersiniz. Yerel SKS'ler de bu sorundan muzdariptir

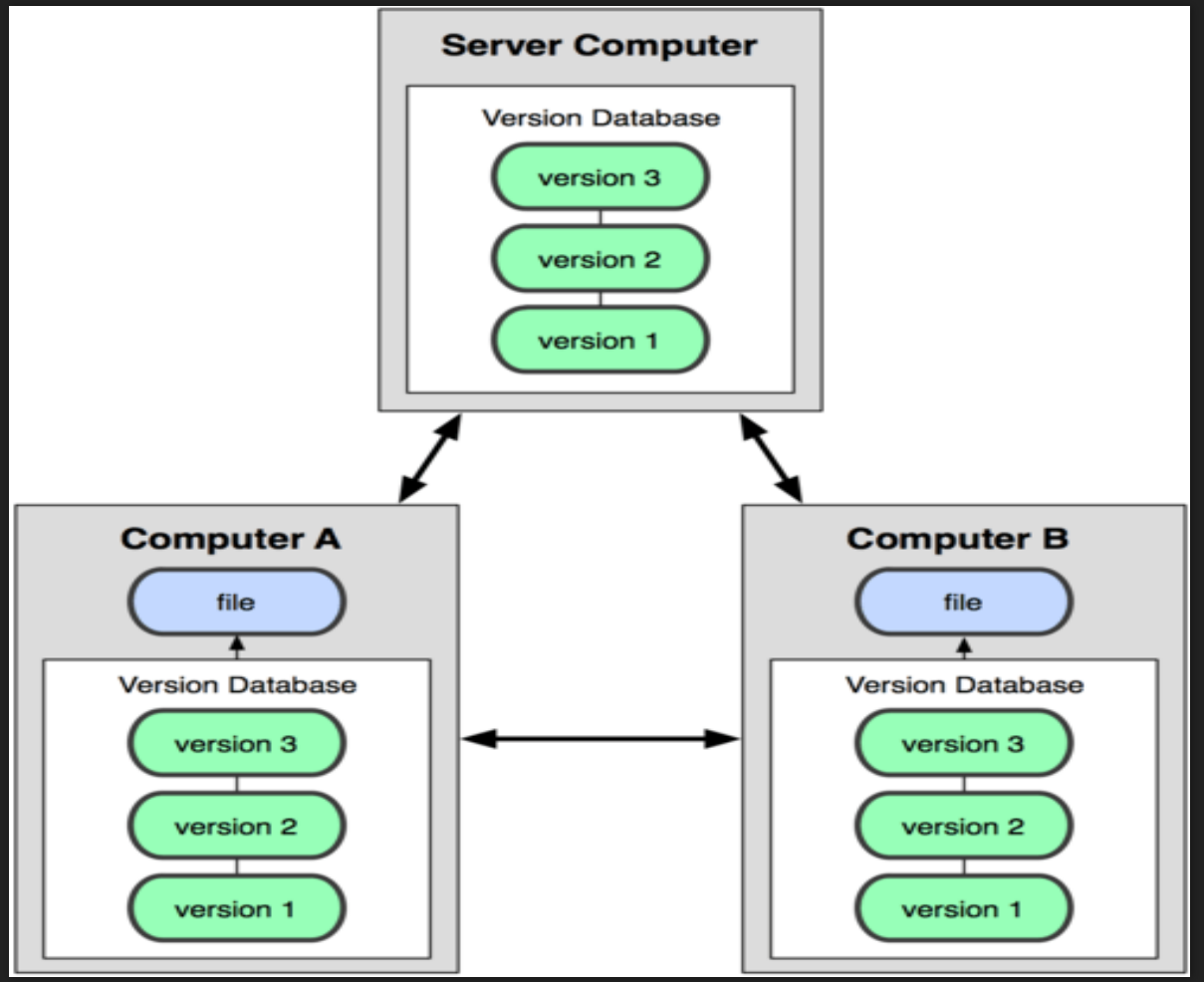
Dağıtık Sürüm Kontrol Sistemleri

- Merkezi olmayan sistemlerde versiyonlama iki kademeli olarak gerçekleşir.
- Merkezi sistemlerde tek depo (repository) ve birçok kullanıcı vardır.
- Merkezi olmayan sistemlerde ise birçok yerel depo (local repository) ve birçok kullanıcı ile birlikte bir adet de uzak depo (remote repository) bulunmaktadır.
- Bu tarz sistemlerde, commit işlemi sadece yerel depoyu etkiler. *

Dağıtık Sürüm Kontrol Sistemleri

- “Push” eylemi, daha önce yerel makinede commit edilmiş. Bütün revizyonları sırayla uzak depoya uygular.
- Uzak depodaki versiyonun yerel depoya indirilmesi işlemine ise merkezi olmayan sistemlerde Pull denir

Dağıtık Sürüm Kontrol Sistemleri



Dağıtık Sürüm Kontrol Sistemleri

- Dahası, bu sistemlerden çoğu birden çok uzak uçbirimdeki yazılım havuzuyla rahatlıkla çalışır, böylece, aynı proje için aynı anda farklı insan topluluklarıyla farklı biçimlerde ortak çalışmalar yürütebilirsiniz.
- Bu, birden çok iş akışı ile çalışabilmenizi sağlar, ki bu merkezi sistemlerde (hiyerarşik modeller gibi) mümkün değildir.

Git’in Kısa Tarihçesi

- Hayattaki pek çok harika şey gibi, Git de bir miktar yaratıcı yıkım ve ateşli tartışmayla başladı
- 2002 yılında, Linux çekirdek projesi, BitKeeper adında tescilli bir DSKS kullanmaya başladı.
- 2005 yılında, Linux çekirdeğini geliştiren toplulukla BitKeeper'ı geliştiren şirket arasındaki ilişki bozuldu ve aracın topluluk tarafından ücretsiz olarak kullanılabilmesi uygulamasına son verildi.
- Bu, Linux geliştirim topluluğunu (ve özellikle Linux'un yaratıcısı olan Linus Torvalds'ı) BitKeeper'ı kullanırken aldıkları derslerden yola çıkarak kendi araçlarını geliştirme konusunda harekete geçirdi.

Git'in Kısa Tarihçesi

- Yeni sistemin hedeflerinden bazıları şunlardı:
  - Hız
  - Basit tasarım
  - Çizgisel olmayan geliştirim için güçlü destek (binlerce paralel dal (*branch*))
  - Bütünüyle dağıtık olma
  - Linux çekirdeği gibi büyük projelerle verimli biçimde başa çıkabilme (hız ve veri boyutu)

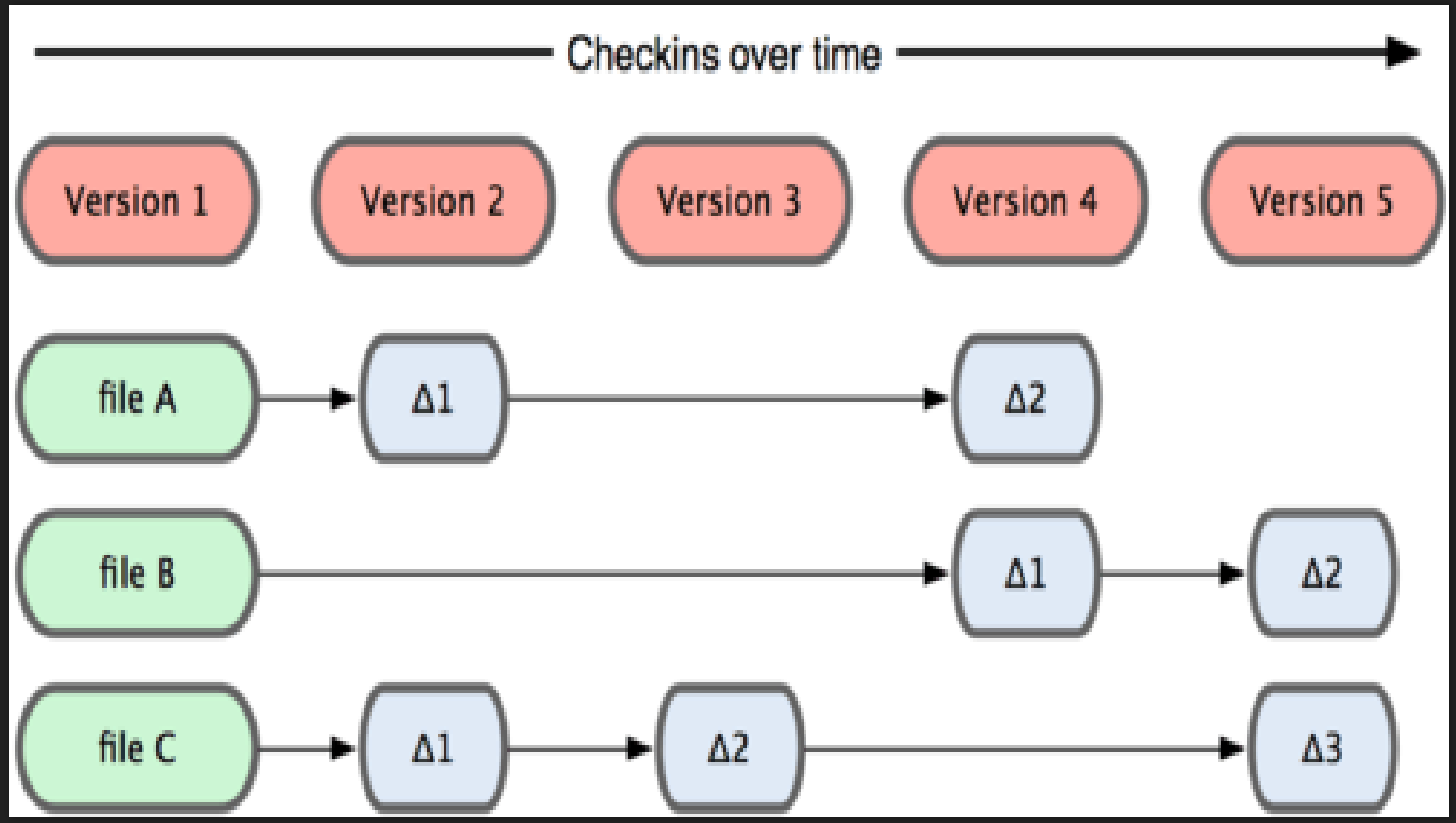
Git'in Kısa Tarihçesi

- Git, inanılmaz ölçüde hızlı, büyük ölçekli projelerde alabildiğine verimli ve çizgisel olmayan geliştirim için inanılmaz bir dallanma (*branching*) sistemine sahip.

SVN vs GİT

- Git, yerel depolar ile de çalıştığından internete bağlı olmadığınız sürede de kendi değişikliklerinizi kademe kademe versiyonlamanız mümkündür.
- Yeni başlayanlar için SVN'i anlamak Git'i anlamaktan biraz daha kolaydır. SVN'in yardım dosyaları daha organizedir.
- Git'te, her yerel depo bir şekilde uzaktakinin yedeği olduğundan veri kaybetmek neredeyse imkansızdır

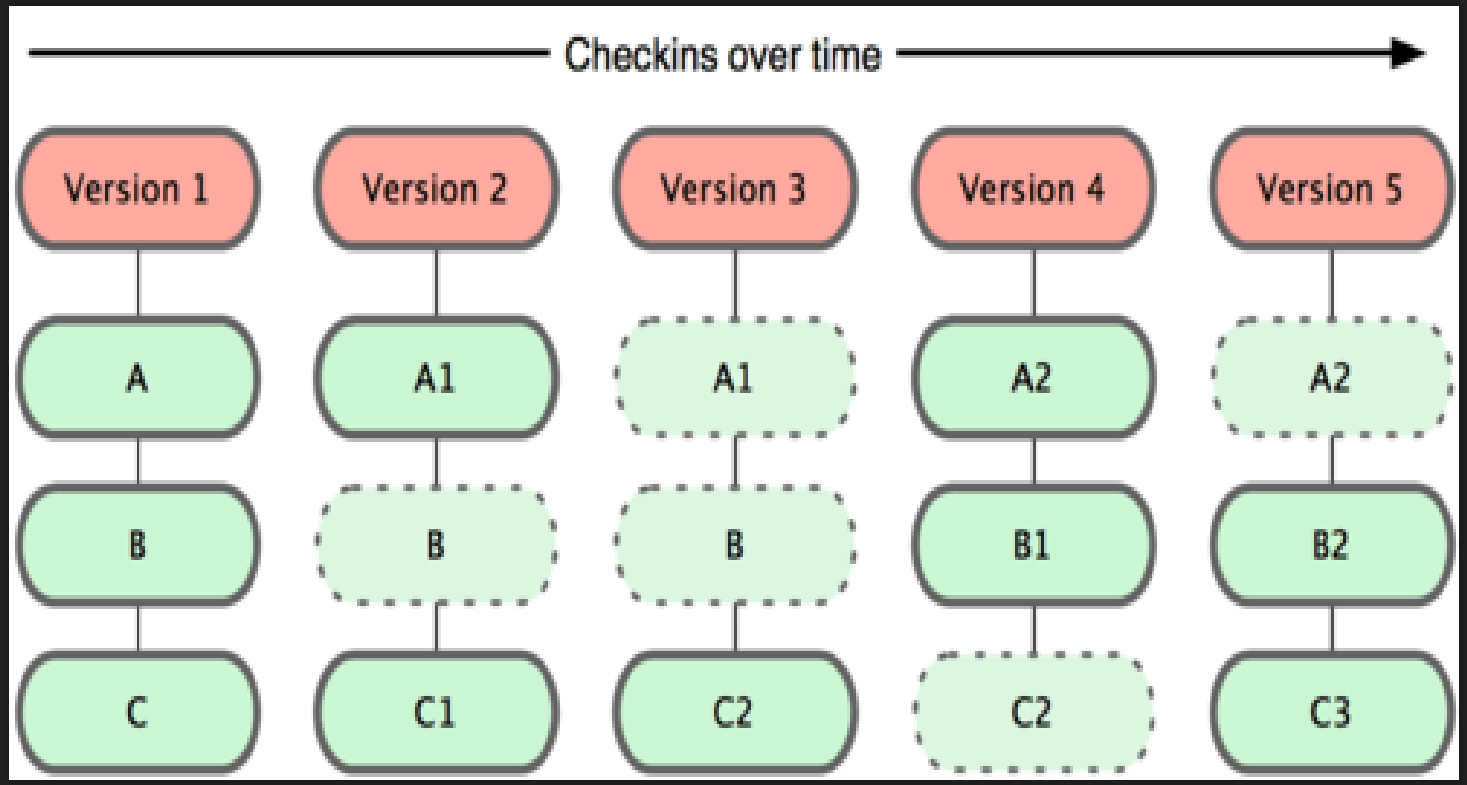
Git'in Temelleri



SVN vs GİT

- Git'in birleştirme işlemi SVN'e göre kat kat daha iyidir. Conflict oluşma sıklığı daha azdır.
- Git, disk alanını SVN'e göre kat kat daha verimli kullanır.
- Git'te uzak sunucuya erişim SVN'e nazaran daha nadir olduğundan versiyonlama işlemi daha hızlı çalışır.

Git'in Temelleri



Github