Git Sürüm Kontrol Sistemi

Çağrı Emer

Nisan, 2016

İçindekiler

1 Giriş

Daha önceki yazımda Subversion'dan bahsetmiş ve Git'e ileriki bir zamanda değineceğimi söylemiştim. Bu yazıda Git'in temel yapısı ve Subversion ile arasındaki farklardan bahsetmeye çalışacağım.

2005 yılında Linus Torvalds tarafından başlatılan ve geçtiğimiz yıllar içerisinde de oldukça gelişen açık kaynak Git sürüm kontrol sistemi, Linux Çekirdeği, Perl, Gnome, PostgreSQL ve Debian gibi pekçok büyük projenin geliştirilmesinde kullanılıyor. Şimdi merkezi şekilde çalışan Subversion'ın aksine, dağıtık bir sürüm kontrol sistemi olan Git'in nasıl çalıştığına bakalım.

2 Blob mu? O da ne?

Git, Subversion gibi dosyaları takip etmekten ziyade içeriği takip eder ve içeriğin SHA1 toplamını veri adı olarak kullanır. Adreslenebilir içerik olarak adlandırılan bu kavram, nesnenin tuttuğu ham veri dışında başka bir şeyden haberdar olmamasını sağladığından, nesnelere verilen referanslar tek yönlüdür. Yani blob adı verilen nesne bir dosyayı ve dosyanın içinde ne olduğunu tutmaktansa direkt olarak dosya içeriğini alır ve SHA1 toplamını isim olarak kullanır. Böylelikle, kriptografik açıdan güçlü olarak addedilen SHA1 algoritması sayesinde her değişikliğin farklı bir isime sahip olacağından emin olunur. Blobları, üzerinde değişiklik yaptığımız belgelere benzetirsek çok da
yanılmış sayılmayız. Yalnız şunu eklemek gerekir ki bu benzetme yalnızca kavramın daha rahat anlaşılması için kullanılmıştır. Bloblar aslında geleneksel
anlamda bizim bildiğimiz belgeler değillerdir.

3 Tree Nesnesi

Blobların içinde tutulduğu dizinler Git altında tree olarak adlandırılırlar. Aynı blob nesneleri gibi, tree nesneleri de sadece tuttukları blobların adlarını bilirler. Tree nesneleri de SHA1 toplamları alındığından, Subversion'da olanın aksine tuttukları verinin SHA1 toplamı değiştiğinde değişirler. Tree'leri bilgisayarımız içinde bulunan dizinlere benzetebiliriz. Yine bloblarda olduğu gibi bu benzetme kavramın sadece daha iyi anlaşılması için kullanılmıştır.

SHA1 toplamlarının alınması birbirlerine oldukça benzeyen fakat küçük değişikliklere sahip olan iki adet büyük boyutlu dosya için ilk başta etkin bir yöntem olarak görünmeyebilir. Fakat Git, bu işi dosya arasındaki farklılıkları tutan delta'lar ile optimize eder ve oldukça yer tasarrufu sağlar.

4 Commit ve Tag'ler

Commit adı verilen nesne bir depoya yapılan değişikliğin tutulduğu veriden ibarettir. Commit'ler aslında tam olarak yapılan değişikliklerin listesini tutmazlar gerçi bunu yapmaları da gerekmez çünkü bir commit ile, içinde bulunduğu tree ve değişiklik yapılan dosya karşılaştırılarak değişiklikler kolayca hesaplanabilir. Commit'i geleneksel bir sistemde en son değişikliği kimin nerede yaptığını tutan bilgi olarak düşünebiliriz.

```
1 1da177e4c3f41524e886b7f1b8a0c1fc7321cac2
```

Git içinde bulundurduğu her nesneyi 40 karakterden oluşan SHA1 toplamı ile ifade ettiğinden dolayı bir dosya ismini okumak insanlar için oldukça zordur. Örneğin, yukarıdaki sayılar aslında Linus Torvalds'ın, Linux Çekirdeği'ne Git üzerinden yaptığı ilk commit'i temsil etmektedir. Dolayısıyla bu problemin çözümü adına tag denen etiketleri kullanıma sunmuştur. Tag'ler bir nesneyi işaret eden kullanıcı dostu isimlerden oluşur. Bu tip nesneler için de dizinlerimizi tuttuğumuz kısımlar üzerine yapıştırdığımız ufak hatırlatıcı notları verebiliriz.

5 Git, SVN'e karşı

Git'in temel mantığını anladıktan sonra sıra şimdi SVN ile arasındaki farklara geldi. Yazının başında da belirttiğimiz üzere bu iki sistemin en büyük farkı birinin dağıtık birinin ise merkezi bir sistem olmasıdır. Git ile çalışmak için her seferinde ana sunucuya bağlanmak zorunda kalmazsanız. Dağıtık olmasının güzel yanı budur. Zira depoyu bir kere yerelinize aldığınız zaman hiç ana depoya
bağlanmaya gerek duymadan yerelde istediğiniz gibi çalışmanıza olanak verir. Subversion ise merkezi bir sistem olduğundan çalışmak için ağ bağlantısına ihtiyacınız vardır. Günümüzde sürekli ağ bağlantısına sahip olmak büyük bir problem teşkil etmeyebilir. Fakat çalıştıracağınız her komutun ağ erişimine ihtiyaç duyması bekleme zamanı olarak size geri dönecektir. Bu açıdan bakıldığında Git, Subversion'a göre oldukça avantajlıdır.

Yine merkezi bir sistem olmadığından Git üzerinde çalışırken veri kaybetme ihtimalimiz Subversion'da çalışırkenkinden daha azdır. Zira dağıtık olması nedeniyle ana deponun kopyaları pek çok istemcide bulunur ve ana depoya zarar gelmesi halinde diğer istemcilerden depo kurtarılabilir. Öte yandan Subversion merkezi bir depoda tutulduğundan, herhangi bir hatada veri kazanımı oldukça zordur ve tutulan yedeklerden geri dönülmesini gerektirebileceğinden son yapılan değişikliklerin tamamen kaybolmasına yol açabilir.

İki sistemin disk kullanımına geldiğimizde avantajın yine Git'ten yana olduğunu görürüz. Subversion her verinizin iki kopyasını tutar. Biri çalıştığınız biri de sunucunuza yaptığınız commit'leri bildirmek için kullanılan referans dosyalardır. Dolayısıyla bu diskte olması gerektiğinden fazla yer kaplamasına neden olur. Sayısal bir örnek verecek olursak Mozilla vakfının on senelik çalışmalarının tutulduğu Subversion deposu 12GB'ın üzerinde yer kaplarken, aynı deponun Git eşleniği 420MB kadar yer kaplamaktadır. Arada otuz kata yakın bir fark vardır. Disk kullanımı bu haldeyken Subversion'ın aslında bir depo içindeki herhangi bir dizini yerelinize indirmeye olanak sağlaması görece avantajdır. Yani yukarıdaki örnek için konuşursak Firefox 3.6'nın kodlarını incelemek için 12GB veriyi yerelinize çekmek zorunda değilsinizdir. İhtiyacınız olanı indirir ve işinizi görürsünüz. Öte yandan Git deposu ile çalışacak olursanız Mozilla'nın 420MB'lik tüm verişini indirmeniz gerekir. Görüldüğü gibi aslında iki sistem burada birbirini dengelemektedir. Birini kısmen çekebilmeniz, diğerinin ise tamamını çekmek zorunda olmanıza rağmen depo boyutunun küçük olması avantajlardır.

Eğer sahipli kod yazıyorsanız Git yararınıza olmayabilir. Zira dağıtık bir sistem olduğundan kullanıcılar bir kere deponun kopyasını aldıklarında isterlerse bunu başkalarıyla paylaşabilirler. Subversion böyle değildir. Herkes merkezi bir depoda çalışmak durumunda olduğundan depoya kimin ne zaman nasıl erişeceği bellidir ve kodun istemsizce dağıtılması konusunda avantaj sağlar. Öte yandan merkezi depolar, yaptığınız her değişikliğin herkes tarafından görülmeden istediğiniz birkaç kişiyle çalışmanızı sağlamak için ekstra erişim kontrol sistemlerine gerek duyarken Git için bir arkadaşınızla istediğiniz commit'leri paylaştığınız bir ortamda ana depoya commit yapmanıza dolaşıyla geliştirmenin erken safhalarındaki kısımları herkesin görmesine izin vermenize gerek kalmaz. Bu aslında yaptığınız geliştirmeyi olgunlaştırabileceğiniz bir alan demektir. Dolayısıyla bir projenin farklı kısımlarında çalışan takımlara sahip olan kullanıcılar için Git kısmen daha avantajlıdır diyebiliriz.

Git'i öğrenmesi Subversion'a göre biraz daha zordur denilebilir. Fakat öğrenme süreci tamamlandıktan sonra Git'in kodu dallara ayırma ve birleştirme konusunda olan eksikliği öğrenmek için geçirilen süreyi tolare edecektir. Subversion üzerinde de kodu dallandırma ve istendiğinde birleştirme işlemleri yapılabilse de bu Git'e göre oldukça hantal kalmakta ve uzaktaki bir sunucuda asıl kodun kopyaları çıkarılarak yapılmaktadır.

Linux altında Git oldukça performanslı çalışmaktadır fakat diğer platformlar için aynı şeyi söylemek pek doğru olmayacaktır. Öte yandan Subversion her platform altında sunucu performansına bağlı olarak öldükça iyi çalışmaktadır. Bunu Subversion projesinin Git'ten daha uzun süredir ortada olmasına bağlamak yanlış olmayacaktır. Zira aynı şey görsel araçlar için de geçerlidir. Subversion'ın daha uzun süredir kullanılıyor olması görsel geliştirme araçlarının Subversion için daha gelişmiş olmasını mantıklı kılar.

Subversion üzerinde geri ve ileriye doğru gitmek dosya versiyonları 1, 2, 3 gibi değiştiğinden oldukça kolay ve içgüdüseldir. Git için bu böyle değildir çünkü isimlendirme için başka bir altyapı kullanır. Öte yandan Git için GitHub gibi sosyal bir kodlama platformu mevcut ve çeşitli kullanımlar için ücretsizken Subversion için bildiğim kadarıyla böyle bir organizasyon ya da şirket yoktur. Dolayısıyla Subversion kullanmak için kendi sunucunuzu kurmanız gerekir.

Yazının geneline bakıldığında, yazarın Git'in daha iyi bir sistem olduğunu düşündüğü yanılgısına kapılabilinir. Halbuki durum böyle değildir. Yukarıda verilenler her kaynakta bulabileceğiniz genel geçer doğruları ifade etmektedir. Şahsi görüşlerim yazının dışında tutulmuştur. Ayrıca bu satırların yazarı olarak ben, Git'i öğrenmek için çok tembel olduğumu bahane ederek Subversion kullanmaktayımdır :) Dolayısıyla kullanacağınız sürüm kontrol sistemini tamamen kendi ihtiyaçlarınız ve öğrenmek için üzerine ayıracağınız vakit belirleyecektir. İki sistemi de kullanan dünya çapında oldukça büyük projeler olması da zaten bu savı destekler niteliktedir.