**Merge Alternatifi Olarak Rebase Kullanımı**

Merge komutu iki branch arasındaki değişiklikleri entegre etmenin en kolay yolu olmakla birlikte tek yol değildir. Rebase komutu da iki branch'ı entegre etmek için kullanılan merge komutuna alternatif bir komuttur. Bu durumda kafanızda "Neden **merge** yerine **rebase** kullanmak isteyelim?" şeklinde bir soru oluşabilir. Bu sorunun cevabını bulmak için önce gelin **merge** komutunun biraz daha iyi anlamaya çalışalım.

**Merge komutuna daha yakın bir bakış**

Git merge işlemini gerçekleştirmeden önce aşağıdaki üç commit'i tespit eder

* **İki branch'in ortak commit'i:** İki branch'in de tarihçesini daha yakından incelediğinizde bu branch'lerin zamanın bir noktasında ortak bir commit'e sahip olduklarını görürüz. Bu anda her iki branch'in de içeriği bire bir aynıdır.
* **Branch'lerin son commit'leri:** Her iki branch için de yapılan son commit'ler

Bu üç commit tespit edildikten sonra Git bu üç commit'i birleştirerek entegrasyonu yapabilir.

**Fast-Forward ve Merge Commit**

Basit bazı durumlarda branch'lerden bir tanesinde herhangi bir değişiklik yapılmamıştır ve bu branch'in yukarıdaki bölümde belirttiğimiz ortak commit'i ve son commit'i aynıdır. Bu durumda merge işlemi çok basitleşir ve git diğer branch'in tüm commit'lerini ortak commit'in üzerine ekleyerek merge işlemini yapar. Bu özel duruma Git terminolojisinde **"Fast-Forward Merge"** denir ve her iki branch'in tarihçesi de ortaktır.

Fakat çoğu zaman her iki branch'de birbirinden bağımsız olarak değişikliğe uğrar ve tarihçe açısından birbirinden uzaklaşırlar. Bu durumda merge işlemini yapmak için Git'in her iki branch arasındaki değişiklikleri içeren otomatik bir commit oluşturması gerekir. Oluşturulan bu commit'e Git terminolojisinde **"Merge Commit"** denir.

**Normal Commitleri ve Merge Commitleri Ayırt Etmek**

Normal commit'leri yazılım geliştiriciler ince eleyip sık dokuyarak oluşturulurlar, diğer yandan Merge Commitler ise Git tarafından otomatik oluşturulurlar. Merge işlemi ile ilgili ayrıntıları daha sonradan incelemek isterseniz her iki branch'in commit tarihçesine ve commit çizelgesine bakmanız gerekir.

**Rebase ile değişiklikleri entegre etmek**

Bazı takımlar iki branch'i yukarıda anlattığımız otomatik merge commit'ler yerine **rebase** ile entegre etmeyi tercih edebilir. Rebase sonrasında projenizin iki farklı branch'i olduğuna dair herhangi bir tarihsel iz oluşmaz.

Gelin şimdi rebase işleminin nasıl yapıldığına bakalım. Örnek senaryomuzda Branch-B'deki değişiklikleri Branch-A'ya entegre edeceğiz. Rebase işlemini **git rebase** komutunu aşağıdaki gibi kullanarak yapıyoruz.

**$ git rebase Branch-B**

Bu komut ile Git öncelikle Branch-A ile Branch-B'nin ortak en son commit'ini bulup ortak commit sonrasında Branch-A'da yapılan diğer tüm commit'leri geri alır. Aslında bu commitler silinmez sadece geçici olarak farklı bir yerde saklanır. Daha sonra Branch-B'deki tüm commitler Branch-A'ya uygulanır. Son aşamada ise Branch-A'nın geçici olarak farklı bir yerde saklanan commit'leri tekrar uygulanır. Bu işlemler sonrasında tüm değişiklikler sanki sadece Branch-A üzerinde gerçekleşmiş gibi görünür.