**Basit Bir Branching Akışı**

Gelin şimdi hep birlikte günlük çalışmanız sırasında kullanabileceğiniz basit bir branching akışını ele alalım. Çalışma senaryomuzun şöyle geliştiğini düşünelim

1. Bir web sitesi üzerinde çalışmaya başladınız
2. Bu siteye yeni bir özellik eklemek için bir branch oluşturdunuz
3. Bu yeni branch üzerinden değişikliklerinizi yapmaya başladınız

Bu sırada web sitesinde bir güvenlik açığı tespit edildiğini bildiren bir email aldınız. Acil olarak bu güvenlik açığını gidermeniz için yapmakta olduğunuz çalışmayı bırakmanız ve bu durumu düzeltmeniz gerekiyor. Böyle bir durumda aşağıdaki adımları takip edebilirsiniz

1. Aktif branch'inizi web sitenizin son stabil versiyonunun bulunduğu **master** branch olarak değiştirdiniz.

**git checkout master**komutunu kullandık

1. Güvenlik açığını giderme çalışmanız için yeni bir branch oluşturdunuz.
   * **git branch loginsorunu** komutunu kullanarak branch oluşturduk ve
   * **git checkout loginsorunu** komutu ile bu branch'i aktif > hale getirdik
2. Güvenlik açığını giderecek değişikliği tamamladınız, testlerinizi yaptınız ve bu değişikliği Staging Area'ya ekleyip sonrasında da commit ettiniz.
   * **git add login.xyz login.html login.css** ile değişiklikleri Staging Area'ya gönderdik
   * **git commit -m "Özel karakter içeren kullanıcı adlarında ortaya çıkan güvenlik sorunu giderildi"** ile değişikliklerimizi commit ettik.
3. **master** branch’imizi aktif hale getirdik.

**git checkout master** komutu ile

1. Commit ettiğiniz değişikliği web sitenizin stabil versiyonunu içeren **master** branch’imize merge ettik.

**git merge loginsorunu**

1. Daha önce üstünde çalışmakta olduğunuz yeni özellik ile ilgili değişiklikleri içeren branch'inizi aktif hale getirerek çalışmanıza kaldığınız yerden devam edebilirsiniz.

**git checkout yeniozellik\_xyz** komutu ile

**Checkout, HEAD ve Working Copy kavramları**

Git'de bir branch otomatik olarak o branch için yaptığınız son commit işlemine bir işaretçi tutar ve hangi dosyaların o branch'e ait olduğunu bilir. Herhangi bir anda bir proje için tek bir branch **aktif** olabilir. Bu branch'e **HEAD** denir ve Working Copy içindeki (Working Copy'yi projenizin yerel diskinizdeki dosyalarının tamamı olarak düşünebilirsiniz) dosyalar aktif olan branch'e yani **HEAD**'e aittir. Diğer branch’lerinizdeki dosyalar diskiniz üzerinde değil Git'in veri tabanında (.git klasörü içinde özel bir formatta) bulunur.

Farklı bir branch'i aktif hale getirmek için **git checkout** komutu kullanılır. Bu durumda Git otomatik olarak sizin için iki şey yapar

1. Aktif hale getirdiğiniz branch'i **HEAD** yapar ve
2. Aktif hale getirdiğiniz branch'e ait dosyaları Git veri tabanınızdan yerel diskinize kopyalar ve önceki branch'e ait dosyaları diskinizden kaldırır. Yani Working Copy'nize yeni branch'e ait olan dosyaları koyar.