Merge Conflicts

Bir merge conflict'i tam olarak nedir, conflict marker'ları (<<<<<<<, =======, >>>>>>>) nasıl okunur, gerçek Java kod örnekleriyle nasıl çözülür, merge ve rebase sırasında aynı mekanik, ve git merge --abort/git rebase --abort ile güvenli iptal.

Orta 30 dk

Son güncelleme: 27 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/merge-conflicts

---

Merge conflict'lerinin iki kez bahsedildiğini zaten gördün — biri "Branches & Merging"te, biri "Rebase & Squash"ta — ve bilinçli olarak bu özel ders için bırakıldı. Burada tam mekaniği ele alıyoruz: Git'in sana tam olarak ne gösterdiği, tam olarak senden ne yapmanı istediği, ve panik yapmak yerine sakince conflict'leri çözmek için tekrarlanabilir bir workflow.

Merge Conflict Nedir?

Bir **merge conflict**, Git iki değişiklik kümesini birleştirmeye çalışırken hangi sürümün doğru olduğuna otomatik olarak karar veremediğinde olur, çünkü her iki taraf da aynı dosyanın aynı satırlarını farklı şekillerde değiştirmiştir. Git, merge'i (ya da rebase'i) yarı yolda durdurur ve senden — bir insandan — karar vermeni ister.

Conflict'ler Neden Olur

Git, yapabildiği her zaman otomatik olarak merge eder — bir takımdaki değişikliklerin çoğu hiç çakışmaz, çünkü farklı dosyalara ya da farklı, örtüşmeyen satırlara dokunurlar. Bir conflict özellikle **her iki branch'in de aynı satırları değiştirmiş olmasını** gerektirir. Aynı sınıfa ilgisiz metotlar ekleyen iki geliştirici genellikle temiz bir şekilde merge olur; aynı metodun *aynı satırını* değiştiren iki geliştirici bir conflict'i tetikleyen şeydir.

Conflict İşaretlerini Anlamak

Bir conflict olduğunda, Git her iki sürümü de işaretlerle sarmalanmış şekilde doğrudan dosyaya yazar:

```
public int calculateDiscount(Order order) {
<<<<<<< HEAD
    return order.getTotal() > 100 ? 10 : 0;
=======
    return order.getTotal() > 100 ? 15 : 5;
>>>>>>> add-loyalty-discount
}
```

Bunu kesin bir şekilde oku:

- `<<<<<<< HEAD`, **mevcut branch'inin sürümünün** başlangıcını işaretler.
- `=======`, iki sürümü ayırır.
- `>>>>>>> add-loyalty-discount`, **gelen branch'in sürümünün** sonunu işaretler, ve o branch'i adlandırır.

Git *neyin* değiştiği konusunda kafası karışık değil — sana iki geçerli seçenek gösteriyor ve doğru sonucu seçmeni (ya da birleştirmeni) istiyor, çünkü her tarafın gerçek niyetini yalnızca sen bilirsin.

Bir Merge Conflict'ini Çözmek

Çözmek, dosyayı tam olarak istediğin kodu içerecek şekilde düzenlemek demektir — işaretleri kaldırıp aralarındaki içeriği seçmek (ya da birleştirmek):

```
public int calculateDiscount(Order order) {
    return order.getTotal() > 100 ? 15 : 5;
}
```

Özel bir "conflict çözme modu" yoktur — yalnızca normal bir metin dosyasını düzenliyorsun. İşaretler tek alışılmadık kısımdır, ve onlar gittikten ve kod doğru olduktan sonra, Git dosyayı çözülmüş olarak kabul eder.

Bir Merge'i Tamamlamak

Her çakışan dosya düzenlendikten ve işaretler gittikten sonra, çözülmüş dosyaları stage'le ve commit et:

```
#!/bin/sh
# Merging a branch, hitting a conflict, resolving it, and completing the merge.

git merge add-loyalty-discount
# Output:
# Auto-merging DiscountCalculator.java
# CONFLICT (content): Merge conflict in DiscountCalculator.java
# Automatic merge failed; fix conflicts and then commit the result.

git status
# Output:
# On branch main
# You have unmerged paths.
#   (fix conflicts and run "git commit")
# Unmerged paths:
#   (use "git add <file>..." to mark resolution)
#         both modified:   DiscountCalculator.java

# DiscountCalculator.java now contains conflict markers:
#
# public int calculateDiscount(Order order) {
# <<<<<<< HEAD
#     return order.getTotal() > 100 ? 10 : 0;
# =======
#     return order.getTotal() > 100 ? 15 : 5;
# >>>>>>> add-loyalty-discount
# }
#
# ... edit the file, keep the correct logic, remove all three marker lines ...

git add DiscountCalculator.java
git commit
# Output:
# [main 3f4a5b6] Merge branch 'add-loyalty-discount'

git status
# Output:
# nothing to commit, working tree clean
```

Buradaki `git add`, her zamankinden biraz farklı bir şey ifade eder — yeni bir değişikliği stage'lemiyor, Git'e "bu dosyanın conflict'i çözüldü" diyor. Ardından gelen commit, merge commit'in kendisidir.

Rebase Sırasında Conflict'ler

Conflict'ler bir rebase sırasında da olabilir — rebase commit'leri tek tek yeniden oynattığı için, sıradaki herhangi bir bireysel commit'te bir conflict oluşabilir, ve Git tam olarak o commit'te duraklar.

Rebase Conflict'lerini Çözmek

Çözüm mekaniği bir merge conflict'iyle aynıdır — dosyayı düzenle, işaretleri kaldır, çözülmüş dosyayı `git add` et — ama ilerlemek için kullanılan komut farklıdır, çünkü bir rebase'in bundan sonra hâlâ yeniden oynatacak commit'leri olabilir:

```
git add ResolvedFile.java
git rebase --continue
```

Sıradaki başka commit'ler de çakışırsa, Git her birinde tekrar durur — rebase bitene kadar aynı çöz-ve-devam-et adımlarını tekrarla.

Bir Merge'i İptal Etmek

Bir conflict beklenenden daha karmaşık çıkarsa, ya da merge'in şu an hiç gerçekleşmemesi gerektiğini fark edersen, onu tamamen iptal edip başlamadan önceki duruma tam olarak dönebilirsin:

```
git merge --abort
```

Bir Rebase'i İptal Etmek

Devam eden bir rebase için karşılığı:

```
git rebase --abort
```

Her iki `--abort` komutu da tamamen güvenlidir — endişelenecek kısmi ya da bozuk bir durum yoktur. Bir conflict, çözümünden emin olmayacak kadar kafa karıştırıcıysa, iptal edip yeniden yaklaşmak (belki çakışan değişikliği yazan kişiyle konuştuktan sonra) her zaman makul bir seçimdir.

Pratik Bir Conflict Çözme Workflow'u

Herhangi bir conflict için sakin, tekrarlanabilir bir süreç:

1. Tam olarak hangi dosyaların çakıştığını görmek için `git status` çalıştır.
2. Her çakışan dosyayı aç ve herhangi bir şeye dokunmadan önce her işaret bloğunun *her iki* tarafını oku.
3. Her tarafın neyi başarmaya çalıştığını anla — yalnızca sözdizimini değil, niyeti.
4. Dosyayı doğru nihai sonuca düzenle, tüm işaretleri kaldırarak.

5. Yapabiliyorsan projenin testlerini çalıştır — derlenen çözülmüş bir conflict, mutlaka *doğru* çözülmüş bir conflict değildir.
6. Her çözülmüş dosyayı `git add` et.
7. `git commit` (merge) ya da `git rebase --continue` (rebase) ile tamamla.

Yaygın Hatalar

- **Commit edilen kodda yanlışlıkla conflict işaretlerini bırakmak** — bu gerçekten olur, ve `main`'e giden bir sözdizimi hatasıdır (ya da daha kötüsü, sessizce yanlış bir mantık). Çözdükten sonra dosyayı her zaman yeniden oku, ve build/testlerinin aceleyle yapılan bir okumanın kaçırabileceğini yakalamasına izin ver.
- **Diğer tarafın değişikliğini neden yaptığını anlamadan bir tarafı seçmek** — bazen "doğru" çözüm ne biri ne diğeridir, ikisinin bir kombinasyonudur.
- **Conflict ortasında paniğe kapılmak** — `--abort` her zaman seni güvenliğe geri götürür; baskı altında kafa karıştırıcı bir conflict'i zorlamana gerek yok.

En İyi Pratikler

- Düzenlemeden önce bir conflict'in her iki tarafını da tam olarak oku — yalnızca tanıdık gelmeyeni silme.
- Çözdükten sonra, merge/rebase'i tamamlamadan önce testleri çalıştır — derlenen bir conflict çözümü mantıksal olarak hâlâ yanlış olabilir.
- Bir conflict gerçekten belirsizse, çakıştığın kişinin değişikliğini yazan kişiyle konuş — genellikle kendi tarafının niyetini hemen bilirler.

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Bir merge conflict, her iki branch de aynı dosyanın aynı satırlarını değiştirdiğinde olur — Git otomatik olarak seçemez, bu yüzden senden ister.
- Conflict işaretleri (`<<<<<<<`, `=======`, `>>>>>>>`), her iki sürümü doğrudan dosyada gösterir; çözmek, onları doğru nihai içeriğe düzenlemek demektir.
- Aynı çöz-ve-devam-et mekaniği hem merge'lerde (`git commit`) hem rebase'lerde (`git rebase --continue`) geçerlidir.
- `git merge --abort` ve `git rebase --abort`, conflict-öncesi duruma dönmek için her zaman güvenli kaçış kapılarıdır.

Cheat Sheet

```
git status                    # hangi dosyaların çakıştığını gör
# ... dosyaları düzenle, <<<<<<< ======= >>>>>>> işaretlerini kaldır ...
git add <resolved-file>       # bir dosyanın conflict'ini çözülmüş olarak işaretle
git commit                    # bir merge'i tamamla
git rebase --continue         # bir rebase'in bir adımını tamamla
git merge --abort             # çakışan bir merge'i tamamen iptal et
git rebase --abort            # çakışan bir rebase'i tamamen iptal et
```

Terimler Sözlüğü

- **Merge conflict** — Git'in iki branch'in aynı satırlardaki değişikliklerini otomatik olarak birleştiremediği durum.
- **Conflict marker** — Git'in çakışan iki sürümü göstermek için eklediği `<<<<<<<` / `=======` / `>>>>>>>` satırları.
- **Resolve (çözmek)** — çakışan bir dosyayı tek, doğru nihai sürüme düzenlemek.