# Stash

git stash ile commit edilmemiş değişiklikleri bir kenara kaydetmek, apply vs pop farkı, çoklu ve isimlendirilmiş stash'ler, git stash drop/clear, ve takip edilmeyen dosyaları stash'lemek. Gerçekçi bir "işi yarıda bırakıp başka göreve geçmek" senaryosu.

**Orta** 25 dk

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

---

Her geliştiricinin başına gelen bir durum: bir özelliğin ortasındasın, kodun henüz derlenmiyor, ve aniden branch değiştirmen gerekiyor — bir production bug'ı geldi, ya da bir takım arkadaşının başka bir branch'te hızlı bir incelemeye ihtiyacı var. Commit etmeye hazır değilsin. `git stash`, tam olarak bunun için var.

## git stash Nedir?

`git stash`, commit edilmemiş değişikliklerini — hem stage edilmiş hem edilmemiş — alır ve ayrı, geçici bir depolama alanına kaydeder, working tree'ni son commit'e uyacak şekilde geri yükler. Bu esasen "commit et, ama gerçekten değil, ve hiçbir branch'te değil" demektir — yaptığın şeyi gerçekten bitirmeden temiz bir working tree'ye geri dönmenin bir yolu.

## Değişiklikleri Stash'lemek

```
git stash
```

Bu kadar — takip edilen her dosyanın commit edilmemiş değişiklikleri kaydedilir, ve working tree'n `HEAD`'e uyacak şekilde geri döner. Artık serbestçe branch değiştirebilirsin, çünkü geçişle çakışacak commit edilmemiş hiçbir şey kalmamıştır.

## git stash list

Stash'ler birikir — aynı anda birden fazlasına sahip olabilirsin. Neyin kaydedildiğini gör:

```
$ git stash list
stash@{0}: WIP on add-password-reset: 8a1b2c3 Add PasswordController
stash@{1}: WIP on main: a1b2c3d Fix typo in README
```

Her giriş bir index (`stash@{0}` en yenisidir), stash'lendiği branch, ve dayandığı commit'i gösterir.

## git stash apply

En son stash'in değişikliklerini working tree'ne geri getir, **stash listesinden kaldırmadan**:

```
git stash apply
```

Aynı stash'i daha sonra tekrar uygulamak istersen, ya da farklı bir branch'e uygulamak istersen `apply`'ı kullan — yıkıcı değildir.

## git stash pop

Daha yaygın gündelik komut — en son stash'i uygula **ve tek adımda listeden kaldır**:

```
git stash pop
```

Bir stash'in içeriğiyle işin bittiğinde, gerçekten istediğin şey neredeyse her zaman `pop`'tur — arkasını temizler.

## Belirli Bir Stash'i Uygulamak

Birden fazla stash'in kaydedildiyse, herhangi birine index ile referans ver:

```
git stash apply stash@{1}
git stash pop stash@{1}
```

## Stash'lere İsim Vermek

Birkaç stash birikince, `WIP on <branch>: <hash> <message>` birbirinden ayırt etmesi zor hale gelebilir. Bir stash oluştururken ona akılda kalıcı bir açıklama ver:

```
git stash push -m "half-finished password strength validator"
```

(`git stash push`, düz `git stash`'in daha açık, modern biçimidir — ikisi de aynı şeyi yapar, ama `push` bir mesaj ve birkaç başka seçenek kabul eder.)

## git stash drop

Belirli bir stash'i uygulamadan sil — o değişikliklere artık ihtiyacın olmadığından emin olduğunda işine yarar:

```
git stash drop stash@{0}
```

## git stash clear

Tüm stash'leri tek seferde sil:

```
git stash clear
```

> Hem `drop` hem `clear`, stash'lenmiş değişiklikleri kalıcı olarak atar — hiçbir onay istemi yoktur. Herhangi birini çalıştırmadan önce emin olduğundan emin ol.

## Takip Edilmeyen Dosyaları Stash'lemek

Varsayılan olarak, `git stash` yalnızca Git'in zaten takip ettiği dosyalardaki değişiklikleri stash'ler — yepyeni, takip edilmeyen dosyalar dokunulmadan bırakılır. Yeni dosyaları da stash'lemek istiyorsan:

```
git stash -u          # takip edilmeyen dosyaları da dahil et
git stash --all       # takip edilmeyen VE ignore edilen dosyaları da dahil et
```

## Gerçek Dünyadan Bir Stash Workflow'u

```
#!/bin/sh
# A developer is mid-way through a Java feature (code doesn't even compile yet)
# when an urgent production bug needs attention on a different branch.

git status
# Output:
# On branch add-password-reset
# Changes not staged for commit:
#         modified:   PasswordValidator.java
#         modified:   PasswordController.java

# Not ready to commit -- the code doesn't compile yet. Stash it aside.
git stash push -m "half-finished password strength validator"
# Output:
# Saved working directory and index state On add-password-reset: half-finished password
strength validator

git status
# Output:
# On branch add-password-reset
# nothing to commit, working tree clean

# Switch to main and fix the urgent bug.
git switch main
git switch -c fix-login-crash
# ... fix the bug ...
git commit -am "Fix NullPointerException on login when session expired"
git switch main
git merge fix-login-crash
git branch -d fix-login-crash

# Back to the feature -- restore exactly where we left off.
git switch add-password-reset
git stash list
# Output:
# stash@{0}: On add-password-reset: half-finished password strength validator

git stash pop
# Output:
# On branch add-password-reset
# Changes not staged for commit:
#         modified:   PasswordValidator.java
#         modified:   PasswordController.java
# Dropped stash@{0} (a1b2c3d4e5f6...)

# Exactly the same uncommitted state as before the interruption.
```

## Yaygın Hatalar

- **Bir stash'in var olduğunu unutmak.** `git stash` yapıp tamamen başka bir şeye geçmek kolaydır, haftalar sonra unutulmuş bir stash yığını keşfetmek üzere — düzenli olarak `git stash list` iyi bir alışkanlıktır.
- **Alışkanlıktan `pop` yerine `apply` kullanmak**, sonra stash listesinin neden büyümeye devam ettiğine şaşırmak — stash'i özellikle yeniden kullanman gerekmiyorsa, genellikle doğru olan `pop`'tur.
- **Takip edilmeyen dosyaların varsayılan olarak stash'lendiğini varsaymak** — `-u` ya da `--all` geçmedikçe değildir.

## En İyi Pratikler

- Birden fazla aktif stash'in olduğu an, stash'lere bir mesaj ver (`git stash push -m "..."`) — gelecekteki sen sana teşekkür edecek.
- Yaygın durum için `pop`'u tercih et; aynı stash'i tekrar istediğini bildiğinde özellikle `apply`'a başvur.
- Stash'i gerçekten geçici olarak ele al — iş bir gün ya da ikiden fazla stash'lenmiş durursa, bu genellikle onun yerine düzgün bir commit (ya da kendi küçük branch'inde bir `git commit --amend`) olması gerektiğinin bir işaretidir.

## Özet, Cheat Sheet ve Terimler Sözlüğü

### Özet

- `git stash`, commit edilmemiş değişiklikleri bir kenara kaydeder ve temiz bir working tree'yi geri yükler, işini commit etmeden görev değiştirmene izin verir.
- `apply`, bir stash'in değişikliklerini geri yükler ve listede tutar; `pop` geri yükler ve kaldırır — `pop` yaygın gündelik seçimdir.
- Stash'ler birikir (`stash@{0}`, `stash@{1}`, ...) ve isimlendirilebilir, listelenebilir, index ile uygulanabilir, tek tek silinebilir ya da tamamen temizlenebilir.
- Takip edilmeyen dosyalar varsayılan olarak bir stash'ten hariç tutulur — dahil etmek için `-u` kullan.

### Cheat Sheet

```
git stash                       # commit edilmemiş değişiklikleri bir kenara kaydet
git stash -u                    # takip edilmeyen dosyaları da dahil et
git stash list                  # kaydedilmiş tüm stash'leri gör
git stash pop                   # en son stash'i geri yükle, ve kaldır
git stash apply stash@{1}       # belirli bir stash'i geri yükle, listede tut
git stash drop stash@{0}        # belirli bir stash'i sil
git stash clear                 # tüm stash'leri sil
```

### Terimler Sözlüğü

- **Stash** — commit edilmemiş değişikliklerin geçici, branch dışı bir kaydı.
- `stash@{N}` — belirli bir kaydedilmiş stash'e referans vermek için kullanılan index (0 = en yeni).