Claude Code CLI: Komutlar ve Kullanım

Terminali kapatıp geri dönmekten context yönetimine, izin modlarından komut keşfetmeye: Claude Code CLI'ını günlük kullanımda etkin kullanmak için hızlı bir başvuru kaynağı -- resmi dokümantasyon ve gerçek bir CLI oturumuyla doğrulanmış.

Orta 22 dk

Son güncelleme: 22 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/claude-code-cli-commands

---

Claude Code CLI: Komutlar ve Kullanım

"Claude Code ile AI Destekli Yazılım Geliştirme" tek bir gerçek oturumun baştan sona anlatısıydı -- analiz, plan, implementasyon, test, review, git. Bu ders farklı bir amaca hizmet ediyor: artık iş akışını bildiğinize göre, günlük kullanımda karşınıza çıkan pratik anları -- terminali kapatıp geri dönmek, context dolduğunda ne yapacağını bilmek, izin modları arasında geçiş yapmak -- hızlıca bulabileceğiniz bir başvuru kaynağı olmak.

> Bu derste anlatılan komut davranışları iki kaynaktan geliyor: resmi Claude Code dokümantasyonu (code.claude.com/docs) ve bu platformun geliştiricisinin gerçek CLI oturumunda -- 22 Ağustos 2026'da, Claude Code v2.1.238, Claude Pro planıyla -- yapılan canlı doğrulama. CLI sürümden sürüme, hatta hesap/plan türüne göre değişebilir; burada değişmez kabul edilmesi gereken tek şey komutların arkasındaki amaç ve iş akışıdır, ekran metinlerinin birebir kendisi değil. Kendi ortamınızda farklı bir şey görürseniz `claude --version` ile sürümünüzü kontrol edin ve resmi dokümantasyona bakın.

Bu Derste Ne Öğreneceğiz?

1. CLI'ı nasıl başlatacağımızı ve etkileşimli mod ile `-p` (print modu) arasındaki farkı.
2. Bir oturumu nasıl sürdüreceğimizi, kapatıp geri döneceğimizi, ya da temiz bir şekilde sıfırlayacağımızı -- session yönetimi.
3. Context (bağlam) dolduğunda ne yapacağımızı -- `/compact` ile `/clear` arasındaki gerçek farkı.
4. İzin modlarını ve Plan Mode'u nasıl kontrol edeceğimizi.
5. Yeni bir komuta ihtiyaç duyduğumuzda onu nasıl keşfedeceğimizi -- ezberlemeden.
6. Günlük kullanımda gerçekten işe yarayan birkaç CLI seçeneğini.

Bu dersin amacı bir komut listesini baştan sona ezberletmek değil: "ihtiyacım olduğunda doğru Claude Code komutunu nasıl bulur, hangi durumda hangisini kullanırım?" sorusuna hızlı bir cevap vermek.

CLI'a Başlamak

"Claude Code ile AI Destekli Yazılım Geliştirme"de gördüğümüz gibi araç, proje dizininde `claude` komutuyla başlatılır. Pratikte en sık kullanılan üç başlangıç şekli:

```
claude                        # etkileşimli oturum, boş
claude "görevi tarif et"      # etkileşimli oturum, ilk görevle
claude -p "görevi tarif et"   # tek seferlik, etkileşimsiz -- yanıtı basıp çıkar
```

Bu dersin odağı ilk ikisi -- etkileşimli terminal iş akışı. `-p` (print modu), etkileşimli bir oturum açmadan çalışan, script ve otomasyon senaryoları için tasarlanmış bir moddur -- örneğin bir CI pipeline'ındaki bir kontrol adımı gibi yerlerde kullanılır. "Etkileşimsiz" olması, hiçbir izin /onay mekanizması olmadığı anlamına gelmez: `--permission-prompt-tool` gibi seçenekler, print modunda bile izin isteklerinin nasıl ele alınacağını tanımlamanızı sağlar. Bu derste derinlemesine girmiyoruz -- yalnızca var olduğunu ve ne için kullanıldığını bilmeniz yeterli.

Bir oturum başladığında terminalin üstünde bir banner, altında ise geçerli izin modunu gösteren bir durum satırı görünür. Bu platformun projesinde gerçekten çalıştırılan bir oturumda (22 Ağustos 2026, v2.1.238, Claude Pro) görünen tam ekran şuydu:

```
Claude Code v2.1.238
Sonnet 5 with medium effort · Claude Pro
~/javaProjects/learning-platform

» auto mode on (shift+tab to cycle)
```

Alttaki satır önemli: oturumun hangi izin modunda olduğunu (`auto` , `plan` , `manual` gibi) ve bunu `shift+tab` ile değiştirebileceğinizi söylüyor. Bu moda "Planlama ve Kontrol: İzin Modları ve Plan Mode" bölümünde tekrar döneceğiz.

Session Yönetimi

Terminali kapatıp tekrar açtığınızda, ya da bambaşka bir işe geçtiğinizde, "şimdi ne yapmalıyım" sorusunun cevabı amaca göre değişir:

- **Aynı terminalde, aynı işe kesintisiz devam ediyorum:** hiçbir şey yapmanıza gerek yok -- konuşma zaten sürüyor.
- **Terminali kapattım, bu dizindeki en son konuşmaya dönmek istiyorum:** `claude --continue` (kısaltması `-c`). Resmi `--help` çıktısındaki tanımı tam olarak şu: "Continue the most recent conversation in the current directory." Argüman almaz, hiçbir seçim ekranı çıkmaz -- doğrudan en sonuncuya döner.

- **Belirli, adını ya da ID'sini bildiğim bir oturuma dönmek istiyorum:** `claude --resume <id>` (kısaltması `-r`). `--help`'in tanımı: "Resume a conversation by session ID, or open interactive picker with optional search term." Yani argümansız çalıştırılırsa (`claude --resume`) bir seçim ekranı açar; bir ID ya da isim verilirse doğrudan o oturuma döner.
- **Oturum içindeyken geçmiş bir konuşmaya dönmek istiyorum:** `/resume` (slash komutu) -- `--resume`'un argümansız haliyle aynı seçim ekranını açar, tek farkı terminalden değil oturumun içinden çağrılmasıdır.
- **Aynı projede yeni, ilgisiz bir işe temiz bir context'le başlamak istiyorum:** `/clear` (takma adları `/reset`, `/new`) -- resmi tanımı "Start a new conversation with empty context." Önceki konuşmayı yok etmez, yalnızca geçerli context'i boşaltıp yeni bir konuşma başlatır; gerektiğinde önceki konuşmaya `/resume` ile geri dönebilirsiniz.
- **Bir oturumu daha sonra kolayca bulmak istiyorum:** ona bir isim vermek -- `-n` / `--name` seçeneği tam olarak bunun için var: "Set a display name for this session (shown in the prompt box, /resume picker, and terminal title)." İsimlendirilmiş bir oturum, aylar sonra `/resume` listesinde "hangi oturum hangi işti" diye tahmin etmenizi gereksiz kılar.

Gerçek Bir Örnek: Kesintiye Uğramış Bir Oturuma Dönmek

"Claude Code ile AI Destekli Yazılım Geliştirme"deki Quiz oturumu sırasında terminalden çıkmak zorunda kalınmıştı. Terminal tekrar açılıp `claude` argümansız çalıştırıldığında, Claude Code şu ipucunu verdi:

```
→ learning-platform git:(master) ✗ claude

Resume this session with:
claude --resume <session-id>
```

Yani `claude` argümansız çalıştırıldığında bu dizinde tamamlanmamış bir oturum olduğunu fark etti ve devam komutunu doğrudan önerdi. Önerilen komut aynen çalıştırıldığında:

```
→ learning-platform git:(master) ✗ claude --resume <session-id>

Resume this session with:
claude --resume <session-id>
```

-- oturum, plan dosyası dahil tüm bağlamıyla kaldığı yerden devam etti (bkz. "Adım 2 -- Plan Mode: Kod Yazmadan Önce Planlamak"da `/plan` ile önizlenen plan). Burada bir ID verildiği için hiçbir seçim ekranı çıkmadığına dikkat edin -- `claude --resume` argümansız çalıştırılsaydı, aşağıda göreceğiniz `/resume` ekranındakine benzer bir seçici açardı.

`/resume` komutunun oturum içinden çağrıldığında açtığı gerçek ekran şöyle görünüyor:

```
Resume session

  Search...

  learning-platform

> enum-quiz-feature
  15 hours ago · master · 911.2KB

Ctrl+A to show all projects · Ctrl+B to only show current branch ·
Space to preview · Ctrl+R to rename · Type to search · Esc to cancel
```

Ekranda iki şey dikkat çekiyor. Birincisi, oturumlar projeye göre gruplanıyor (`learning-platform`) ve içinde isimlendirilmiş bir oturum (`enum-quiz-feature`) görünüyor -- yani "Session Yönetimi"nde bahsedilen isimlendirme alışkanlığı gerçekten işe yarıyor. İkincisi, arama/filtreleme ve dal (branch) bazlı filtreleme gibi kısayollar var -- birden fazla projede paralel çalışırken bu ekranı hızlı kullanmanın anahtarı bunlar.

Context Yönetimi

Context (bağlam penceresi) doldukça ne yapmanız gerektiği de amaca göre değişir:

- **Konuşma uzadı, yanıtlar yavaşladı ya da maliyet arttı, ama aynı işe devam ediyorum:** `/compact` -- konuşmayı özetleyip aynı konuşmada devam eder. İsteğe bağlı bir talimat da verebilirsiniz: `/compact Kod örneklerine ve API kullanımına odaklan` gibi.
- **Bambaşka, ilgisiz bir işe geçiyorum:** `/compact` değil `/clear`. Resmi dokümantasyonun vurguladığı fark şu: `/compact` önce tüm geçmişi okuyup özetlediği için kendisi de büyük ve pahalı bir istektir; `/clear` ise sıfırdan başladığı için bedavadır. İlgisiz bir işe geçerken özetlemeye gerek yok -- temiz başlamak hem daha ucuz hem de eski işin kalıntısını hiç taşımaz.
- **Context'in ne kadar dolu olduğunu görmek istiyorum:** `/context` (`/context all` daha ayrıntılı bir döküm verir) -- neyin ne kadar yer kapladığını gösterir.
- **Otomatik özetleme ne zaman devreye giriyor:** context, oturumun "auto-compact" eşiğine yaklaştığında Claude Code kendiliğinden özetler; `CLAUDE.md` içine bir "Compact instructions" bölümü ekleyerek özetleme sırasında neyin korunacağını projeye özgü olarak belirtebilirsiniz -- "Proje Bağlamı: CLAUDE.md'nin Rolü"nde gördüğümüz otomatik-okunan bağlam dosyasının, context yönetimine özel küçük bir uzantısı.

Karar kuralı basit: **ilgisiz iş -> `/clear`, aynı iş ama context doluyor -> `/compact`.**

> `/compact` , konuşmanın tamamını okuyup özetleyen büyük bir istektir -- context hâlâ rahatça sığıyorsa gereksiz yere çalıştırmak, çözdüğünden fazla maliyete yol açabilir; asıl faydası context gerçekten dolmaya yaklaştığında ortaya çıkar. Bambaşka bir işe geçerken de zaten `/compact` değil `/clear` kullanmanız gerektiğini unutmayın.

Planlama ve Kontrol: İzin Modları ve Plan Mode

"Claude Code ile AI Destekli Yazılım Geliştirme"deki Plan Mode anlatısı tek bir oturumun akışını izliyordu; burada mekaniğin kendisine -- izin modları arasında nasıl geçiş yapılır, Plan Mode'a nasıl girilir -- daha yakından bakıyoruz.

Claude Code'da birkaç izin modu var: her okuma-dışı eylem için soru soran en dikkatli mod, dosya değişikliklerini otomatik onaylayan bir mod, yalnızca keşif yapıp değişiklik yapmayan Plan Mode, ve bir sınıflandırıcının eylemleri sizin yerinize gözden geçirdiği "auto" modu gibi. Oturum içindeyken `shift+tab` bu modlar arasında döngüsel olarak geçiş yapar; "CLI'a Başlamak"ta gördüğümüz `» auto mode on` gibi durum satırı, o an hangi modda olduğunuzu her zaman gösterir. Döngünün kendisi sabit değildir: resmi dokümantasyona göre başlangıç modu `auto` ise ilk `shift+tab` sizi `manual`'e (varsayılan mod) döndürür, oradan sonra döngü `manual -> acceptEdits -> plan -> ...` şeklinde devam eder ve -- hesabınızda etkinse -- `auto` ile isteğe bağlı modlar (`bypassPermissions`, `dontAsk` gibi) döngüye eklenir. Kısacası: `shift+tab` ile mevcut izin modları arasında geçiş yaparsınız, ama hangi modların ve hangi sırada göründüğü sürümünüze/ hesabınıza/ayarlarınıza göre değişebilir -- durum satırı her zaman en güvenilir kaynaktır.

> Bu platformun geliştiricisinin gerçek hesabında (Claude Pro, v2.1.238, `~/.claude/settings.json`'da izin moduna dair özel bir ayar yokken) yeni bir oturum "auto mode on" ile başladı -- bu, resmi dokümantasyonun "Pro/Max/Team planlarında yerleşik varsayılan artık auto" iddiasıyla örtüşüyor, ama "Claude Code ile AI Destekli Yazılım Geliştirme"deki dosya-dosya manuel onay deneyiminden farklı görünebilir. Çelişki yok: genel oturum modu ile bir planı onaylarken seçtiğiniz yürütme tarzı ("Ready to code?" ekranındaki "manually approve edits" seçeneği) iki ayrı kontrol noktasıdır -- o derste bilinçli olarak ikincisi seçilmişti. Kendi ortamınızda hangi modda olduğunuzu varsaymak yerine durum satırına bakın, özellikle riskli bir görevdeyseniz.

Plan Mode'a girmenin iki yolu resmi dokümantasyonda geçiyor: oturum içinde `shift+tab` ile mod döngüsünde `plan`'a gelmek, ya da doğrudan `/plan [görev tanımı]` yazarak (örn. `/plan enum konusuna quiz ekle`) hem moda geçmek hem görevi belirtmek. İkisinin kendi ortamınızda birebir nasıl davrandığı, bu tür etkileşimli detayların sürümler arası en sık değiştiği noktalardan biri olduğu için, `/help` ile ya da doğrudan deneyerek doğrulamanız en sağlam yöntemdir.

"Planı Onaylamadan Önce Okumak: Gerçek Bir Hata Örneği"nde gördüğümüz migration numarası hatası, Plan Mode'un neden yalnızca bir onay ekranı değil, gerçek bir doğrulama fırsatı olduğunun kanıtıydı -- planı okumadan "Yes" demek, tam olarak o hatayı gözden kaçırırdı.

Komutları Keşfetmek

Yeni bir şeye ihtiyaç duyduğunuzda komutu ezberlemiş olmanız gerekmez -- keşfetmeniz yeterli:

- Oturum içinde boş bir `/` yazmak, filtrelenebilir bir komut listesi açar -- yazmaya devam ettikçe liste daralır.
- `/help`, yerleşik komutların bir özetini gösterir.
- Terminalden, oturum dışında `claude --help` çalıştırmak, CLI flag'lerinin ve alt komutların (`claude mcp`, `claude doctor`, `claude auth` gibi) bir listesini verir.

Güncel ve tam bir referans için ilk durak her zaman resmi Claude Code dokümantasyonu (code.claude.com/docs) olmalı -- komut listeleri, bu dersin kendisinin de gösterdiği gibi, sürümler arasında en hızlı değişen kısımdır.

Faydalı CLI Seçenekleri

`claude --help` gerçekten iyi bir başlangıç noktası, ama **tam bir referans değil**. Bu dersi hazırlarken gerçek `--help` çıktısını (60'tan fazla flag) resmi CLI reference sayfasıyla karşılaştırdık: resmi belgede yer alıp `--help` çıktısında görünmeyen, daha uzmanlaşmış birçok flag var (örneğin sistem promptunu dosyadan ekleyen bir seçenek, ya da agent-takım davranışına özel seçenekler). Doğru zihinsel model şu: **`--help`, günlük kullanım için temel bir keşif aracı; resmi CLI reference ise tam ve güncel referans.**

Günlük kullanımda gerçekten işe yarayan birkaç seçenek (`--help`'in gerçek çıktısından, kısaltılmadan):

- `--model <model>` -- "Model for the current session. Provide an alias for the latest model (e.g. 'fable', 'opus', or 'sonnet') or a model's full name." Oturum içinde `/model` ile de değiştirilebilir.
- `--add-dir <directories...>` -- "Additional directories to allow tool access to." Birden fazla proje/modülle aynı anda çalışırken bağlamı genişletmenin yolu.
- `-n, --name <name>` -- "Set a display name for this session (shown in the prompt box, /resume picker, and terminal title)." "Gerçek Bir Örnek: Kesintiye Uğramış Bir Oturuma Dönmek"teki `enum-quiz-feature` ismi tam olarak bu şekilde verilmiş bir isim.
- `-v, --version` -- sürüm numarasını basar. Bu ders boyunca vurguladığımız gibi, bir davranışın "hangi sürümde doğru olduğunu" bilmenin ilk adımı budur.
- `claude doctor` (bir alt komut, flag değil) -- kurulumunuzun sağlığını kontrol eder.

Son bir not: `git status`, `git diff`, `mvn`, `npm` gibi komutlar Claude Code'un kendi komut seti **değildir** -- bunlar sıradan terminal komutlarıdır. Claude Code, sizin adınıza bunları bash aracıyla çalıştırabilir ("Adım 6 -- Git: Değişiklikleri Gözden Geçirip Commit Etmek"te olduğu gibi), ama bu onları birer Claude Code komutu yapmaz -- `claude --help` çıktısında da bu yüzden hiç yer almazlar.

Best Practices

- Terminali kapattıktan sonra aynı işe dönerken `claude --continue`, belirli/adlandırılmış bir oturuma dönerken `claude --resume` ya da `/resume` kullanmak -- bkz. "Session Yönetimi".

- Sık dönülecek bir oturuma erkenden bir isim vermek, aylar sonra `/resume` listesinde tahmin yürütmemek için -- bkz. "Gerçek Bir Örnek: Kesintiye Uğramış Bir Oturuma Dönmek".
- İlgisiz bir işe geçerken `/clear`, aynı işte context doluyorsa `/compact` kullanmak -- ikisini birbirinin yerine koymamak -- bkz. "Context Yönetimi".
- Bir oturumun hangi izin modunda olduğunu varsaymak yerine durum satırına bakmak, özellikle riskli bir görevdeyken -- bkz. "Planlama ve Kontrol: İzin Modları ve Plan Mode".
- Yeni bir komuta ihtiyaç duyduğunuzda onu ezberlemeye çalışmak yerine `/help` ya da boş `/` ile keşfetmek -- bkz. "Komutları Keşfetmek".
- Bir davranışın hangi sürümde doğru olduğunu bilmek için `claude --version`'ı kontrol etmeyi alışkanlık hâline getirmek -- bkz. "Faydalı CLI Seçenekleri".

Yaygın Hatalar

- **`--continue` ile `--resume`'u karıştırmak.** `--continue` argüman almaz ve yalnızca o dizindeki en son konuşmaya döner; belirli/adlandırılmış bir oturuma dönmek için `--resume` (gerekirse ID/isimle) kullanmanız gerekir -- bkz. "Session Yönetimi".
- **Bambaşka bir işe geçerken `/compact` çalıştırmak.** Bu, gereksiz yere pahalı bir isteğe (tüm geçmişi özetlemek) yol açar -- ilgisiz bir işte doğru komut, bedava olan `/clear`'dır -- bkz. "Context Yönetimi".
- **`claude --help` çıktısını CLI'ın tam referansı sanmak.** Bu ders için gerçek `--help` çıktısı, resmi CLI reference sayfasındaki bazı (uzmanlaşmış) flag'leri içermiyordu -- `--help` bir keşif aracıdır, tam liste için resmi belgelere bakmak gerekir -- bkz. "Faydalı CLI Seçenekleri".
- **Bir oturumun izin modunu varsaymak.** "Auto mode" bir hesapta varsayılan olabilirken başka bir hesapta/sürümde farklı olabilir -- durum satırını kontrol etmeden riskli bir göreve girmek, "Claude Code ile AI Destekli Yazılım Geliştirme"deki güvenlik disiplinini baltalar -- bkz. "Planlama ve Kontrol: İzin Modları ve Plan Mode".

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Bu ders "Claude Code ile AI Destekli Yazılım Geliştirme"deki tek-oturum anlatısından farklı bir amaca hizmet eder: komutları ezberletmek değil, "ihtiyacım olduğunda doğru komutu nasıl bulur, hangi durumda hangisini kullanırım?" sorusuna hızlı bir cevap vermek.
- Session yönetimi amaca göre değişir: aynı dizinde en sona dönmek için `--continue`, belirli /adlandırılmış bir oturuma dönmek için `--resume` ya da `/resume`, boş context'le yeni bir konuşma başlatmak için `/clear`.

- Context yönetiminde `/compact` ve `/clear` birbirinin yerine geçmez: ilgisiz işe geçerken `/clear` (bedava), aynı işte context doluyorsa `/compact` (geçmişi özetler, kendisi de bir maliyettir).
- İzin modu (auto/manual/plan) `shift+tab` ile döngüsel değişir ve durum satırında görünür; bu, bir planı onaylarken "Ready to code?" ekranında seçtiğiniz yürütme tarzından ayrı bir kontrol noktasıdır.
- `claude --help` temel bir keşif aracıdır, CLI'ın tam ve güncel referansı değil -- tam referans için resmi Claude Code dokümantasyonuna bakılmalı.
- `git status`, `git diff`, `mvn`, `npm` gibi komutlar Claude Code'un kendi komut seti değildir -- Claude Code bunları sizin adınıza çalıştırabilir, ama bunlar sıradan terminal komutlarıdır.

Cheat Sheet

- Başlatma: `claude` (etkileşimli) / `claude "görev"` (görevle) / `claude -p "görev"` (tek seferlik, etkileşimsiz).
- Aynı dizinde en son konuşmaya dön: `claude --continue` (`-c`).
- Belirli/adlandırılmış bir oturuma dön: `claude --resume <id>` (`-r`) ya da oturum içinde `/resume`.
- Temiz başla (yeni conversation, boş context): `/clear` (`/reset`, `/new`).
- Oturuma isim ver: `-n` / `--name <isim>`.
- Context'i özetle (aynı işe devam): `/compact [talimat]`.
- Context kullanımını gör: `/context` (`/context all`).
- İzin modunu değiştir: `shift+tab` -- döngüsel, ama sabit değil; hangi modların/hangi sırada göründüğü başlangıç modunuza ve hesap/sürüm ayarlarınıza göre değişir (durum satırına bakın).
- Plan Mode'a doğrudan gir: `/plan [görev tanımı]`.
- Komut keşfet: boş `/`, ya da `/help`.
- Temel CLI seçeneklerini keşfet: `claude --help` (tam referans için resmi dokümantasyon).
- Sürüm kontrolü: `claude --version` (`-v`).

Terimler Sözlüğü

- **Session (oturum):** bir `claude` çalıştırmasının başladığı andan kapandığı (ya da `/clear` ile sıfırlandığı) ana kadar süren, kendi geçmişine ve bağlamına sahip tek bir konuşma birimi.
- **Context (bağlam) penceresi:** bir oturumun o ana kadarki tüm mesaj/araç geçmişinden, modele her istekte birlikte gönderilen kısım -- doldukça yanıt kalitesi ve maliyet etkilenir.
- **Compaction (özetleme):** context dolmaya yaklaştığında (otomatik ya da `/compact` ile elle) geçmiş konuşmanın özetlenip yer açılması işlemi -- konuşma sıfırlanmaz, yalnızca özetlenir.

- **İzin modu (permission mode):** bir oturumun dosya değişikliği/komut çalıştırma önerilerine ne kadar otomatik onay vereceğini belirleyen genel ayar (`manual` , `acceptEdits` , `plan` , `auto` gibi) -- `shift+tab` ile döngüsel değişir, durum satırında görünür.
- **Plan Mode:** izin modlarından biri; Claude Code'un herhangi bir dosyayı değiştirmeden önce bir plan ürettiği ve onay beklediği mod.
- **Print modu (`-p`):** Claude Code'un etkileşimli oturum açmadan, tek bir isteği yanıtlayıp çıktığı, script/otomasyon senaryoları için tasarlanmış çalıştırma biçimi.