PRACTICAL GUIDE / 2026.11

Token kullanımını azaltma rehberi

Claude / Gemini / ChatGPT

Sohbet, Claude Code, Codex, Gemini CLI ve Antigravity

- Daha az gereksiz bağlam
- Daha az yeniden deneme
- Ölçülebilir tüketim ve maliyet

Türkçe sürüm | 25 sayfa
Resmî kaynaklar kontrol edildi: 17 Kasım 2026

Okuma haritası ve hızlı başlangıç

Bu rehber; Claude ve Claude Code, ChatGPT ve Codex, Gemini sohbet ve Google'ın kodlama araçları için hazırlanmıştır. İki dildeki sürüm aynı kapsamı, örnekleri ve kaynakları içerir. API bölümleri, kendi entegrasyonunu geliştiren kullanıcılar için eklenmiştir.

Bölüm	Sayfa
Temeller, öncelikler ve istem tasarımı	3-6
Claude, ChatGPT ve Gemini sohbet	7-9
Kodlama akışı ve araçlara özel uygulamalar	10-14
Talimat dosyaları, araçlar ve API maliyeti	15-18
Belgeler, istem şablonları ve ölçüm	19-21
Sorun giderme ve uygulama planı	22-23
Tıklanabilir resmî kaynaklar	24-25

İlk 15 dakikada yapabilecekleriniz

- Tek bir sonuç belirleyin: örneğin "üç seçenekli karar tablosu" veya "şu testi geçiren düzeltme".
- Yalnızca gerekli dosyaları, veri aralığını ve hata örneğini seçin.
- Çıktı uzunluğunu ve biçimini yazın; aynı bilgiyi üç farklı biçimde istemeyin.
- Alakasız yeni bir iş için yeni sohbet açın; aynı işte geçmişi tekrar yapıştırmayın.
- Kodlama aracında kullanım ekranını kontrol edin; model, çaba düzeyi ve araçları kaydedin.

> Yöntem: Ürün davranışları numaralı resmî kaynaklarla desteklenir. Şablonlar, başlangıç aralıkları ve iş akışları bu rehberin önerileridir; garantili tasarruf oranları değildir. Kaynak kontrol tarihi: 17 Kasım 2026.

Abonelik türünüz, kurum ayarlarınız ve istemci sürümünüz özellikleri değiştirebilir. Komut örnekleri belirtilen CLI içindir; sohbet kutusunda veya her IDE eklentisinde çalıştıkları varsayılmamalıdır.

Token, bağlam, kota ve maliyet

Token, modelin metni ve diğer girdileri işlerken kullandığı birimdir. Bir kelime birden çok token olabilir. Dil, modelin tokenizer'ı, kod, sayılar ve noktalama sonucu değiştirir. İngilizce için kullanılan yaklaşık karakter/kelime oranlarını Türkçeye doğrudan uygulamayın. [01]

Kavram	Neyi ifade eder?	Nasıl azaltılır?
Girdi	Talimat, seçilen geçmiş, dosyalar, araç sonuçları	Gereksiz bağlamı çıkarın
Çıktı	Yanıt, üretilen kod ve araç çağrıları	Gerekli teslimatı tanımlayın
Akıl yürütme	Bazı modellerin görünmeyen düşünme kullanımı	Uygun model/çaba seçin
Bağlam penceresi	Bir istekte kullanılabilen çalışma alanı	Özetleyin veya yeni oturum açın
Kota / kredi	Ürünün kullanım hakkını ölçme biçimi	Ürünün kendi ekranını izleyin

Üç farklı hedefi karıştırmayın

- **Daha az token:** daha küçük girdi, daha kısa çıktı ve daha az tekrar.
- **Daha düşük API faturası:** token sayısına ek olarak model fiyatı, önbellek ve araç ücretleri de önemlidir.
- **Aboneliği daha uzun kullanmak:** sağlayıcının kota kurallarına bağlıdır; token azalması aynı oranda kota tasarrufu garantilemez.

Girdi yaklaşık olarak "sistem talimatı + proje kuralları + ilgili geçmiş + getirilen içerik + yeni mesaj" birleşimidir. Araç kullanan bir ajan tek kullanıcı mesajında birçok model çağrısı yapabilir. Bu yüzden mesaj sayısı, toplam işi veya maliyeti tek başına açıklamaz.

> "Kısa cevap ver" görünür çıktıyı sınırlamaya yardımcı olur. Modele gönderilen geçmişi veya görünmeyen akıl yürütmeyi otomatik olarak küçültmez. OpenAI akıl yürütme tokenlarının API'de çıktı olarak ücretlendirildiğini açıklar. [24]

En etkili tasarruf sırası

Bir istemden birkaç kelime silmeden önce işin toplam akışına bakın. Yanlış dosyanın okunması, belirsiz kapsam ve başarısız denemeler genellikle küçük ifade farklarından daha büyük bir maliyet alanıdır. Aşağıdaki sıra bir başlangıç önerisidir; kendi verinizle doğrulayın.

Öncelik	Yapılacak iş	Dikkat
1	Doğru hedef ve kabul ölçütü yazın	Eksik gereksinim yeniden iş üretir
2	İlgili dosya, sayfa ve satırları seçin	Kritik bağımlılıkları kesmeyin
3	İlgisiz görevleri ayırın	Devir özeti hazırlayın
4	Tekrarlı okuma ve araç çağrılarını azaltın	Güncellik gerektiğinde yeniden okuyun
5	Uygun model ve çaba kullanın	Hata oranını birlikte izleyin
6	Çıktı biçimini daraltın	Kanıt ve doğrulamayı koruyun
7	API önbelleği ve toplu işlem değerlendirin	Token azalması ile fiyat indirimini ayırın

Modeli göreve göre seçin

- Basit dönüşüm, başlık üretimi, sınıflandırma: hesabınızda bulunan hızlı/ekonomik seçenekle başlayın.
- Sınırlı hata düzeltme, birkaç dosyalı geliştirme: dengeli model ve normal çaba deneyin.
- Mimari, karmaşık hata ve güvenlik açısından önemli değişiklik: daha güçlü model ve gereken düşünme derinliğine geçin.

Ekonomik model başarısız olursa aynı belirsiz istemi defalarca yeniden üretmeyin. İlk denemenin neden başarısız olduğunu kısaca belirleyin; eksik kanıtı ekleyin veya modeli yükseltin. Daha güçlü model, tekrarları azalttığında tamamlanmış iş başına daha ucuz olabilir.

> Başarı ölçütü: kabul edilen sonuç başına toplam tüketim. En kısa istem veya en düşük tek çağrı maliyeti, tek başına iyi bir optimizasyon hedefi değildir.

Kısa ama eksiksiz istem yazma

İyi bir istemin beş parçası vardır: **hedef, gerekli bağlam, sınırlar, çıktı biçimi ve kabul ölçütü**. Her görevde uzun bir rol tanımı veya çok sayıda örnek gerekmez. Başlangıç istemini anlam kaybettirmeden kısaltın; ayrıntıyı karar vermeyi etkilediği yerde koruyun. Açıklık ve yeterli bağlam, ChatGPT'nin resmî önerileriyle de uyumludur. [11]

Zayıf başlangıç

```
Bu projeyi incele, her şeyi iyileştir ve detaylı açıkla.
```

Daha verimli başlangıç

```
Hedef: /checkout sayfasındaki çift gönderimi düzelt.
Kanıt: Hızlı çift tıklamada iki POST oluşuyor.
Kapsam: checkout bileşeni ve ilgili test; API değişmesin.
Kabul: Tek POST; hata sonrası yeniden deneme çalışsın.
Çıktı: Değişiklik özeti, test sonucu, kalan risk; en çok 150 kelime.
```

Takip mesajını da daraltın

- "Baştan yaz" yerine: "İkinci paragrafı daha sade yaz; diğer bölümleri koru."
- "Hepsini tekrar anlat" yerine: "Önceki öneriye göre yalnız değişen kararları ve gerekçelerini ver."
- "Daha iyi yap" yerine: "İki alternatifi kurulum süresi ve bakım yüküyle karşılaştır; önerini seç."

İlişkili soruları aynı mesajda gruplayabilirsiniz: "Aynı veriyle özet, üç risk ve bir öneri üret." Ancak bağımsız on işi tek büyük isteme doldurmak, öncelik kaybına ve pahalı düzeltmelere yol açabilir. Ortak girdisi ve ortak kabul ölçütü olan işleri birleştirin.

> Önerilen başlangıç: kısa sohbet yanıtında 100-200 kelime veya 3-5 madde. Bu bir ürün sınırı değildir. Yeterli açıklama gerektiren işlerde bütçeyi artırın; aşırı kısa yanıt yüzünden yeni tur açmak tasarrufu tersine çevirebilir.

Sohbeti yönetme ve devir özeti

Aynı karar üzerinde ilerliyorsanız mevcut sohbeti sürdürmek yararlıdır. Konu, veri kümesi veya hedef değiştiyse yeni sohbet açmak daha temiz bir başlangıç sağlar. Her beş mesajda bir sıfırlama gibi sabit kurallar kullanmayın; korumanız gereken bağlam ile taşıdığınız gereksiz geçmişi karşılaştırın.

Durum	Tercih edilen hareket
Aynı belgenin belirli bölümünü düzeltme	Mevcut sohbette yalnız değişikliği söyleyin
Yeni müşteri, yeni depo veya farklı konu	Yeni sohbet ve kısa gerekli arka plan
Uzun iş sürüyor; eski araç çıktıları birikti	Desteklenen özetleme veya elle devir özeti
Önceki sonuçtaki tek yanlış varsayım	Varsayımı düzeltin; her şeyi tekrar göndermeyin
Kesin ifade / sayı / kaynak kaybolabilir	Özet yerine özgün kanıtı ayrıca koruyun

Kopyalanabilir devir özeti isteği

```
Bu işe yeni sohbette devam edeceğim. En çok 250 kelimede:
1. Amaç ve kabul ölçütleri
2. Kesinleşmiş kararlar ve kısıtlar
3. İlgili dosya/sayfa/veri aralığı ve sürüm
4. Yapılanlar, test sonucu, açık sorunlar
5. Sıradaki tek adım
Kesin hata metinlerini ve kritik değerleri aynen koru.
Gizli bilgi, eski loglar ve vazgeçilmiş seçenekleri ekleme.
```

Özeti okuyup düzeltin; sonra yeni sohbete yalnız bu özeti ve gerekli kanıtı taşıyın. Dosya adı tek başına yeterli olmayabilir: yeni ortamın o dosyaya erişebildiğini doğrulayın. Kompakt özetler, kaynağın tamamının yerine geçmez.

> Özetleme de token tüketebilir ve ayrıntı kaybettirebilir. Sık aralıklarla özetlemek yerine doğal iş aşamalarında kullanın. Yeni sohbet veya geçmiş temizleme, daha önce tüketilmiş kotayı geri getirmez.

Claude sohbet için uygulama

Claude kullanımını mesaj uzunluğu, ekler, sohbet geçmişi, model, çaba ve araçlı işlemler etkiler. Anthropic, Projects içeriğinin yeniden kullanımında önbellek avantajı olduğunu; önbelleğin hareketsizlikten sonra sona erebildiğini belirtir. Bu avantajı sabit bir yüzde veya sınırsız kullanım olarak yorumlamayın. [02]

Belgeler ve yazı işleri için akış

- Tek bir müşteri/konu için Project oluşturun; yalnız sürekli başvurulacak belgeleri ekleyin.
- Project talimatlarını hedef kitle, dil, ton ve beklenen çıktı biçimiyle sınırlayın.
- Tek seferlik uzun yönergeleri her sohbette çalışan kalıcı kurallara dönüştürmeyin.
- Önce gerekli bölümü inceletin: “Sadece 3. bölümdeki varsayımları değerlendir.”
- Uzun metin için önce bölüm planını ve bir örnek bölümü netleştirin; sonra kalanını üretin.
- Sonraki turda tüm belgeyi tekrar üretmek yerine değişen paragrafları isteyin.

Örnek istem

```
Project içindeki teklif-v3 belgesini kullan.
Yalnız kapsam, teslim tarihi ve hariç tutulan işleri incele.
Çıktı: En çok 5 belirsizlik; her biri için bölüm adı ve
önerilen tek cümlelik düzeltme. Yeni teklif yazma.
```

Çalışma “bir raporun düzenlenmesi”nden “başka müşteriye yeni rapor”a döndüğünde yeni sohbet açın. Kalıcı kaynakları Project içinde tutmak, arayüzde tekrar yüklemeyi azaltabilir; modelin kullanacağı bağlamın ücretsiz veya sıfır token olduğu anlamına gelmez.

Kontrol noktası

Göreve başlamadan plan/model kullanım ekranını not edin. İş bittikten sonra kullanılan model, ek sayısı, araçlı aşamalar ve yeniden deneme sayısıyla birlikte değerlendirin. Yalnız yanıtın kaç kelime olduğuna bakarak token tüketimini tahmin etmeyin.

> Bu sayfadaki sohbet önerileri Claude Code komutları değildir. `/compact`, `/context` ve `/usage` gibi CLI komutlarını claude.ai sohbetinde evrensel denetim komutları olarak kullanmayın.

ChatGPT sohbet için uygulama

ChatGPT Projects, ilgili sohbetleri, dosyaları ve talimatları bir araya getirir. Proje belleği ve dosya erişimi ayarlara ve plana bağlıdır. Yeni sohbet açmak, Project içinde kullanılabilen diğer bağlamın tamamen yok olduğunu garanti etmez. Projects kullanımını garanti edilmiş API önbellek indirimiyle eşitlemeyin. [10]

Günlük kullanım düzeni

- Rutin bilgi dönüştürme veya kısa taslak için mevcut hızlı seçeneği kullanın; karmaşık karar için uygun düşünme moduna geçin. Seçenekler hesabınıza göre değişebilir.
- Her göreve “ayrıntılı düşün ve uzun açıkla” gibi kalıcı talimat eklemeyin. Yanıt biçimini ihtiyaca göre belirleyin.
- Web araştırmasını güncellik veya kaynak gerektiren sorularda kullanın; size verilen metnin basit düzenlemesi için gereksiz arama istemeyin.
- Tablo ve hesaplama için ilgili sayfa, sütun, tarih aralığı ve beklenen hesabı söyleyin.
- Kod çıktısı gerektiğinde tek bir çalışır örnek isteyin; aynı turda uzun ders, beş alternatif ve tam proje istemeyin.

Örnek istem

```
Ekli satış tablosunda yalnız 2026 Q3 ve Q4 verisini kullan.
Önce veri eksiklerini belirt. Ardından ciro farkını ve
yüzde değişimi hesapla. Çıktı: 1 küçük tablo ve
en çok 3 yorum. Hesapta kullanılan sütunları belirt.
```

Tüketim ölçeri yoksa güvenilir bir “kalan token” sayısı uydurmayın ve modelin kendi tahminini fatura olarak kabul etmeyin. Plan bildirimlerini, seçili modeli ve tamamlanan iş sayısını kaydedin. Dosya/araç içeren işler için tüketimin yalnız görünür mesaj sayısıyla belirlenmediğini varsayan bir çalışma düzeni kullanın.

> ChatGPT sohbet, Codex CLI ve API farklı yüzeylerdir. `/status` veya API’deki `max_output_tokens` ayarı, normal sohbet kutusuna yazılınca ürün kontrolüne dönüşmez. Codex için sayfa 12’ye bakın.

Gemini sohbet için uygulama

Google, Gemini Apps sınırlarının hesaplama tabanlı olduğunu; istem karmaşıklığı, model/özellik seçimi ve sohbet uzunluğundan etkilendiğini belirtir. Bu nedenle daha kısa mesajlar her planda aynı oranda ek kullanım sağlamaz. Kişisel hesap belgesindeki kuralları iş/okul hesabına otomatik uygulamayın. [12]

Gems ve tekrar eden işler

Gems, belirli işler için yeniden kullanılabilen talimatlar ve bağlam oluşturmanıza yardımcı olur. Google'ın önerdiği öğeler rol, görev, bağlam ve biçimdir. Bir Gem oluşturmak, tüketimin sıfır olduğu veya her isteğin daha az token kullandığı anlamına gelmez. [13]

- Gem talimatına sık kullanılan kısa iş kurallarını koyun; tek seferlik büyük veri dökümlerini koymayın.
- Drive veya ek dosyası üzerinden çalışıyorsanız doğru belge/sürüm ve ilgili bölümü adlandırın. Dosyaya erişim ve uygun paylaşım gerektiğini kontrol edin.
- Sadece verilen kaynakları özetlemek istiyorsanız bunu yazın; güncel araştırma gerekiyorsa arama aralığını ve karar sorusunu belirleyin.
- Uzun PDF için tek bir turda özet, çeviri, sınav, sunum ve rapor istemek yerine öncelikli teslimatı seçin.

Örnek istem

```
Bu Gem'in yazım kurallarını uygula. Ekli raporda yalnız
Yönetici Özeti ve Sonuç bölümlerini kullan.
Hedef kitle: teknik olmayan yönetici.
Çıktı: 120 kelimelik özet + kaynak bölüm adları.
Kaynakta olmayan sayıları veya nedenleri ekleme.
```

Aynı veriyi farklı biçimde tekrar sunarken "önceki sonuçla aynı sayıları koru, yalnız biçimi değiştir" diyebilirsiniz. Yeni konu açıldığında gerekli bilgiyi kısa bir devir özetiyle taşıyın; önceki sohbetin tamamına otomatik erişim olduğunu varsaymayın.

> Gemini sohbet ile Google'ın kodlama ürünleri ayrı ele alınmalıdır. 2026'daki Code Assist / Gemini CLI geçişinin ayrıntıları ve Antigravity önerileri sayfa 13-14'tedir. [14]

Kodlama ajanlarında ortak akış

Kodlama araçlarında büyük tüketim, yalnız yazılan koddan gelmez: depo keşfi, okunan dosyalar, terminal çıktıları, araç şemaları, hata döngüleri ve uzun geçmiş de etkilidir. Başlangıçta hedefi, çalıştırma biçimini ve doğrulama ölçütünü belirtmek, gereksiz keşfi azaltmaya yardımcı olur.

Keşfet > değiştir > doğrula > bitir

- **Keşfet:** Önce dosya adları ve sembolleri ara; sonra ilgili dosyaları oku. Tüm depoyu yazdırma.
- **Daralt:** Hatanın girdisini, beklenen/gerçek sonucu ve yeniden üretme adımlarını ver.
- **Planla:** Birkaç dosyayı etkileyen işte kısa dosya/değişiklik planı iste; küçük düzenlemede fazladan plan turu ekleme.
- **Değiştir:** En küçük tutarlı düzeltmeyi yap; ilgisiz yeniden düzenlemeyi aynı işe ekleme.
- **Doğrula:** Davranışı ölçen ilgili testleri çalıştır; sonuç gerektiriyorsa kapsamı genişlet.
- **Bitir:** Kabul ölçütleri sağlandıysa özet, kanıt ve kalan riskle dur. Süresiz “biraz daha iyileştir” döngüsünden kaçın.

Dar kapsamlı salt okunur komut örnekleri

```
rg --files -g "*checkout*" -g "!node_modules" -g "!dist"
rg -n "submitOrder|isSubmitting" src tests
git diff --stat
git diff -- src/checkout.tsx
```

Örnek yolları ve sembolleri projenize uyarlayın. Test komutlarını tahmin ettirmek yerine README veya paket betiklerinden bulun. Büyük test/log çıktısını yerel dosyada tutup başarısız test adı, ilk ilgili hata ve gerekli satırları modele gönderin.

> Çıktıyı kısaltmak, test çalıştırmaktan vazgeçmek değildir. Başarısız doğrulamayı saklamayın; kritik hata ayrıntılarını bütçe uğruna atmayın. Başarısız sonucu fark etmeden kapatılan iş, sonraki turda daha pahalıya dönebilir.

Claude Code için uygulama

Güncel maliyet belgesi, oturum tokenları ve plan kullanımını görmek için `/usage` komutunu kullanır. Eski sürümlerde ve bazı yardım sayfalarında `/cost` görebilirsiniz; kurulu sürümün yardımını kontrol edin. `/context` bağlamı, `/model` mevcut model seçeneklerini incelemeye yarar. [03]

Amaç	CLI komutu / yaklaşım
Kullanımı görmek	/usage
Bağlamı incelemek	/context
Aynı uzun işi özetlemek	/compact ve korunacak bilgiyi belirtme
İlgisiz işe geçmek	/clear
Modeli göreve uydurmak	/model

Oturumdaki dolar tahminini abonelik faturası sanmayın. Abonelikte plan göstergelerini; API'de sağlayıcının gerçek faturalama ekranını esas alın. Kullanımı temizlemek, daha önce tüketilmiş hakkı iade etmez. [03]

Proje talimatını hafif tutun

`CLAUDE.md` içinde çalıştırma/test komutları, önemli yollar ve gerçek kısıtlar bulunsun. Güncel Claude Code belgesi `AGENTS.md` desteğini de açıklar; davranışı sürümünüzde doğrulayın ve aynı kuralları iki dosyada çoğaltmayın. [04]

Yalnız bazı işlerde gereken uzun prosedürleri bir skill veya görev belgesine taşıyın; ana dosyaya tamamını gömmeyin. Skills, yeniden kullanılabilir görev yönergeleri içindir; içeriği çağrıldığında yine bağlama girebilir. [05]

Örnek kompaktlaştırma isteği

```
/compact Preserve acceptance criteria, changed file paths,
exact failing test names, key decisions and the next step.
```

> Pratik uygulama: işe başlarken kullanım ve bağlamı gözleyin; bitince aynı ekranları tekrar kontrol edin. Görünür yanıt kısa olsa bile araştırma, araç sonuçları ve düşünme tüketimi yüksek kalabilir. Bir komut tanınmıyorsa başka üründeki eşini varsaymayın.

Codex için uygulama

Codex CLI'de `/status` kalan kullanım sınırlarını incelemek için kullanılabilir. Uygun gösterge ve paneller istemciye göre değişir. OpenAI, gereksiz bağlamı, kaynak miktarını, büyük talimat dosyalarını ve kullanılmayan MCP sunucularını azaltmayı önerir. [08]

Amaç	Codex CLI
Kullanım sınırlarını incelemek	/status
Uzun sohbeti özetlemek	/compact
Yeni sohbet açmak	/new
Alternatif bir dal denemek	/fork; geçmişi kopyalar

`/compact` önceki turları kısa özete dönüştürür. `/new` yeni sohbet başlatır. `/fork` mevcut sohbetten bir kopya oluşturduğundan geçmişi sıfırlayan bir tasarruf komutu sayılmamalıdır. Bu komutlar masaüstü veya IDE'nin her sürümünde aynı biçimde sunulmayabilir. [07]

AGENTS.md ve görev çerçevesi

Codex proje talimatlarını hiyerarşik olarak okur; üst düzey kurallar ve çalışma dizinine daha yakın kurallar birlikte etkili olabilir. Dosyayı sırf sınır büyük diye büyütmeyin. Genel varsayımları kısa tutup bileşene özgü kuralları uygun alt dizine yerleştirin. [09]

```
Görev: İade tutarı yuvarlama hatasını düzelt.
Başlangıç: src/refund ve tests/refund.
Kabul: 0.01 fark kaybolsun; para birimi davranışı korunsun.
Önce ilgili kuralları ve testi kontrol et.
İlgisiz dosyaları değiştirme. Sonuç: düzeltme + test kanıtı.
```

Küçük ve iyi tanımlı işi doğrudan yürütün. Belirsiz veya büyük işte kısa bir planla dosya kapsamını netleştirin. İşin sonunu kabul ölçütleriyle tanımlamak, durmadan ek özellik veya yeniden düzenleme üretilmesini önlemeye yardımcı olur. [06]

> Uygulama ayarındaki düşünme çabasını ve model seçimini ihtiyaca göre kullanın. "En yüksek" ayarını her işe kalıcı varsayılan yapmayın. Daha hızlı hizmet seçeneğinin daha az token veya daha düşük fiyat anlamına geldiğini varsaymayın.

Gemini CLI ve Code Assist: sürüm ayrımı

Güncellik notu: Google'ın geçiş belgesine göre 18 Haziran 2026'dan itibaren bireysel Code Assist, Google AI Pro ve Ultra katmanlarında eski IDE eklentileri ve Gemini CLI için Google ile giriş üzerinden hizmet sona erdi; etkilenen kullanıcılar Antigravity'ye yönlendiriliyor. Bu, tüm Gemini API kullanımının sona erdiği anlamına gelmez. [14]

9 Ekim 2026'dan itibaren yeni Code Assist Standard/Enterprise aboneliği satışı da durduruldu. Mevcut sözleşmeler için ayrı destek ve yenileme takvimi var. Hesabınızın hangi ürün/sözleşme üzerinden çalıştığını belirlemeden eski "ücretsiz günlük istek" sayılarıyla bütçe yapmayın. [15]

Gemini CLI kullanan uygun kurulumlar için

Amaç	Gemini CLI komutu
Oturum / model kullanımını görmek	/stats veya /stats model
Geçmişi özetlemek	/compress
Yeni oturuma başlamak	/clear
Yüklenen yönergeleri görmek	/memory show

`/clear` etkin konuşma bağlamını temizler; `Ctrl+L` yalnız görünümü temizler. `/compress` geçmişi özetle değiştirir. Komutları Antigravity'de aynı anlamla varsaymayın. [16]

`GEMINI.md` tekrar eden yönergeleri saklar; `/memory show` yüklenen talimatları incelemeye yarar. Çok büyük global kurallar her işte gereksiz bağlam yaratabilir. [31]

`.geminiignore`, onu destekleyen araçlarda dosya kapsamını daraltabilir; her erişim yolunu engelleyen güvenlik duvarı değildir. Üretilmiş çıktıları dışlayın, gerekli kaynak/test dosyalarını yanlışlıkla dışlamayın. [17]

> Gemini CLI önbellek belgesi API anahtarı ve Vertex AI kurulumlarıyla OAuth kurulumlarını ayırır. `/stats`te cache görünmesi, normal Gemini sohbet kotası için aynı indirimin geçerli olduğunu kanıtlamaz. Eski belge tarihi ile güncel kimlik doğrulama politikasını birlikte kontrol edin. [18]

Güncel Google kodlama akışı: Antigravity

Google'ın güncel yönlendirmesi nedeniyle Antigravity'yi ayrıca ele almak gerekir. Kurulu aracınızın Antigravity CLI mi, IDE eklentisi mi yoksa eski Gemini CLI mı olduğunu belirleyin; kullanım raporlarını ve komutları bu yüzeye göre okuyun. [14]

Kontrol	Antigravity CLI
Etkin bağlam ve kategoriler	/context
Akıl yürütme çabası	/effort; desteklenen modellerde
AI Premium kredi bakiyesi / tüketim geçmişi	/credits
İlgisiz işte yeni sohbet	/clear; diğer adı /new

`/context` kullanıcı mesajı, ajan yanıtı, araçlar ve yüklenen özelleştirmeler gibi bağlam kategorilerini gösterir. Eski turlar otomatik checkpoint özetlerine dönüştürülebilir. Bu ekranı toplam fatura yerine etkin çalışma alanını incelemek için kullanın. [19]

`/effort` desteklenen modele göre çabayı ayarlar. Etkileşimli kullanımda `/effort low`, basit işler için denenebilir; karmaşık işlerde daha yüksek çaba gerekebilir. Başsız çalışmadaki parametreler farklıdır; CLI belgesini kontrol edin. [20]

`/credits`, kredi bakiyesi ve tüketim geçmişini gösterir; kredi sayısını ham token sayısıyla eşitlemeyin. Bakiye raporu ile belirli bir işin bağlam kullanımını ayrı kaydedin. [21]

Geçişte talimatları sadeleştirin

Geçiş belgesi mevcut `GEMINI.md` ve `AGENTS.md` kurallarının kullanılabildiğini, proje skill yolunun `.agents/skills/` olduğunu belirtir. Taşımadan önce eski ve yeni dosyalardaki mükerrer yönergeleri ayıklayın; sırf geçiş yapıldı diye tüm eski eklentileri etkinleştirmeyin. [22]

> Başlangıç önerisi: bir küçük hata düzeltmesini tek ajanla, yalnız gerekli araçlarla tamamlayın. Sonra aynı kabul ölçütünü koruyarak çaba veya araç sayısından yalnız birini değiştirin. Güncel kullanıcı hesabınızda ölçmediğiniz bir tasarruf yüzdesini varsaymayın.

Kalıcı talimat dosyalarını küçültme

Talimatı dosyaya taşımak, elle tekrar yazmayı azaltır; token maliyetini ortadan kaldırmaz. Dosyanın tamamı veya ilgili bölümü oturumlara yüklenebilir. İyi bir talimat dosyası, ajanın doğru davranmasını sağlayan kısa ve güncel bir çalışma sözleşmesidir.

Araç	Yaygın dosya	Tasarruf yaklaşımı
Claude Code	CLAUDE.md; güncel AGENTS.md desteği	Tek yetkili kural seti; koşullu ayrıntı [04]
Codex	AGENTS.md	Kök kısa, alt dizin kuralları yerel [09]
Gemini CLI	GEMINI.md	Global ve projeye özel kuralları ayır [31]
Antigravity CLI	GEMINI.md / AGENTS.md	Geçişte tekrarları temizle [22]

Uyarlanabilir kısa içerik örneği

```
# Proje çalışma kuralları
- Amaç: Ödeme arayüzü ve ilgili servis.
- İlgili yollar: src/payments, tests/payments.
- Test: [depoda doğrulanmış komut].
- Para hesabında mevcut yuvarlama kuralını koru.
- Genel API'yi değiştireceksen önce etkiyi belirt.
- Tamamlanma: ilgili testler ve değişiklik incelemesi.
- Son yanıt: değişiklik, doğrulama, kalan risk.
```

- **Tutun:** gerçek test komutları, kritik mimari sınırlar, sık yaşanan hatayı engelleyen özel kural.
- **Çıkarın:** eski oturum günlüğü, aynı kuralın üç tekrarı, genel motivasyon metni, geçersiz dosya yolları.
- **İsteğe bağlı yükleyin:** nadiren gereken dağıtım prosedürü, uzun stil rehberi, özel veri göçü adımları.

Bir sayfalık dosya bile çelişkiliyse pahalı olabilir. “Her zaman tüm testleri çalıştır” ile “yalnız ilgili testi çalıştır” gibi kuralları tek bir açık ilkeye dönüştürün. Ancak güvenlik ve veri bütünlüğü gereksinimlerini sırf kısa olsun diye silmeyin.

> Öneri: küçük depoda kısa bir kuralla başlayın ve yalnız tekrar eden gerçek sorunlar için yeni madde ekleyin. İçe aktarma veya dosyaya link vermenin her üründe tembel yükleme sağladığını varsaymayın; etkin bağlamı kontrol edin.

Araçlar, MCP, loglar ve çoklu ajanlar

Araç tanımları, çağrı argümanları ve sonuçları bağlama katkı yapabilir. Bazı sistemler araçları gerektiğinde yükler; bazıları daha geniş tanımları baştan sunar. “Bir MCP sunucusu tam olarak şu kadar token harcar” gibi evrensel sabitler kullanmayın. Etkin bağlamı ve gerçek görev tüketimini ölçün.

Daha verimli araç sözleşmesi

- Aramaya klasör, tarih aralığı, dosya türü veya sonuç sayısı sınırı verin.
- Tüm kayıt yerine yalnız gerekli alanları isteyin: id, durum, tarih ve hata nedeni.
- Büyük veriyi yerelde filtreleyin/toplayın; modele karar için gereken küçük özeti gönderin.
- Araç sonuçlarını kararlı biçimde düzenleyin; aynı veriyi farklı sırayla tekrar döndürmeyin.
- Kullanılmayan bağlantı ve araçları o görevde kapatmayı değerlendirin; doğru araca erişimi kaldırıp elle kopyalama döngüsü yaratmayın.

Çoklu ajan ne zaman mantıklı?

Paralel çalışma süreyi kısaltabilir; toplam tokenı düşürmesi şart değildir. Her ajan talimat ve bağlam taşıyabilir, koordinasyon mesajları üretir ve bulguları birleştirmek gerekir. Tek dosyalık düzeltme için birden fazla ajan çoğu zaman gereksiz bir başlangıçtır.

Uygun bölünebilir iş	Maliyetli bölünme
Bağımsız modüller ve ayrı kabul ölçütleri	Aynı dosyayı üç ajana baştan okutmak
Dar kapsamlı uzman incelemesi	Her ajana tüm sohbeti kopyalamak
Büyük çıktıyı okuyup kısa kanıt sunma	Ana ajana tüm alt ajan loglarını taşımak

> Öneri: alt görev verilecekse yalnız amaç, ilgili dosyalar, sınırlar ve istenen sonuç biçimini taşıyın. Sonuçta bulgu + dosya/satır + kanıt + belirsizlik isteyin. Zaman kazancı ile toplam tüketimi ayrı raporlayın.

Güvenlik için gerekli kontrolleri ve yetki sınırlarını kapatmak bir token optimizasyonu değildir. Hedef, işi bitirmek için gereksiz adımları ve çıktıları azaltmaktır.

API: önbellek ve kullanım raporları

Bu bölüm kendi API entegrasyonunuz içindir. Sağlayıcının API ayarı, aynı markanın tüketici sohbetine yazılan talimatla etkinleştirilemez. Önbellek genellikle tekrar kullanılan girdinin maliyetini veya gecikmesini azaltır; girdinin anlamsal uzunluğunu ve bağlam ihtiyacını ortadan kaldırmaz.

Sağlayıcı	Pratik uygulama	Kontrol
OpenAI	Sabit içeriği öne, değişken görevi sona koyun; modelin güncel cache koşullarını kullanın. [23]	Girdi, cached input ve çıktı ayrımı
Claude	Tekrar kullanılan kararlı içeriğe uygun cache_control uygulayın; yazma/okuma ücretlerini ayrı değerlendirin. [28]	input_tokens + cache_creation_input_tokens + cache_read_input_tokens
Gemini	Örtük önbellek fırsatlarını kullanın; API ve modelin açık önbellek desteğini ayrıca doğrulayın. [26]	Önbellek ve diğer kullanım kategorileri

Güncel Gemini belgesi, Interactions API ve Gemini 4 için örtük önbellek yaklaşımını açıklar; açık önbellek nesnelerinin burada desteklendiğini varsaymayın. Daha eski model/endpoint örneklerini doğrudan taşımayın. [26]

Ölçümün doğru yeri

- İstek öncesi sayım: beklenen girdi boyutunu kontrol etmek için; son toplam fatura değildir.
- Yanıt kullanım nesnesi: gerçek istek tüketimini incelemek için; alanlar API ve sürüme göre değişir.
- Fatura/kullanım paneli: araç, cache ve hizmet koşulları dahil parasal hesabı doğrulamak için.

Claude token-counting uç noktası ve Gemini `count_tokens` yöntemi, büyük girdiyi göndermeden boyutu değerlendirmeye yardımcı olur. Tek bir sağlayıcının tokenizer sonucunu diğerine kesin ölçüm olarak uygulamayın. [25][27]

> Önbellek için gereksiz metin ekleyip minimum eşiğe ulaşmaya çalışmayın. İlk yazma, süre dolması, yeniden oluşturma ve varsa saklama ücretlerini hesaba katın. “Cache hit” görmek tek başına daha ucuz bir iş akışını kanıtlamaz.

API bütçesi ve sayısal örnek

Fiyatlar modele, hizmet katmanına, endpoint'e, önbelleğe ve bağlam boyutuna göre değişebilir. Bu nedenle aşağıdaki sayılar tamamen varsayımsaldır; hiçbir sağlayıcının güncel fiyatı değildir. Kendi hesabınızda gerçek birim fiyatları ve kullanım alanlarını kullanın.

```
C = (U × Pu + R × Pr + Wc × Pw + O × Po) / 1.000.000
    + araç ücretleri + varsa saklama / diğer ücretler
U: önbelleksiz girdi; R: cache okuma; Wc: cache yazma
O: sağlayıcının faturalandırdığı toplam çıktı
```

Kategorileri çifte saymayın. Özellikle akıl yürütme toplam çıktının alt kalemiyse onu yeniden eklemeyin. Önbellek yazma ücretinin ayrı mı yoksa girdi içinde mi raporlandığını sağlayıcınızın belgesinden kontrol edin.

Varsayımsal tek istek	Önce	Sonra
Önbelleksiz girdi U	40.000	12.000
Çıktı O	4.000	1.500
Fiyat: girdi $2 / M, çıktı $8 / M	Aynı	Aynı
Hesaplanan maliyet	$0,112	$0,036
Toplam token (bu basit örnekte)	44.000	13.500

Bu senaryoda maliyet azalması yaklaşık %67,9; token azalması yaklaşık %69,3'tür. Aynı oran değillerdir. Örnek cache, ek araç ücreti ve görünmeyen diğer kalemleri içermez; "daha kısa yanıt = her zaman aynı tasarruf" sonucu çıkarılamaz.

Bütçeyi iş akışına koyun

- İstek/iş başına tahmini token ve maliyet sınırı belirleyin; taşma durumunu açıkça yönetin.
- Çıktı tavanını gerekli tamamlamaya yetecek düzeyde tutun; kesilmiş yanıtın tekrar üretilmesi ek maliyet yaratabilir.
- Tekrarlı ağ hatası için sınırlı yeniden deneme ve yinelenen işi önleyen kimlik kullanın.
- Acil olmayan bağımsız API işleri için desteklenen Batch seçeneklerini değerlendirin; bu bir token sayısı azaltma tekniği değil, fiyat/zaman tercihidir. [29]

> Bütçe aşıldığında kaliteyi gizlice düşürmek yerine kısmi sonucu ve kalan işi kaydedin. Gerçek abonelik kotalarını bu varsayımsal dolar hesabına dönüştürmeyin.

PDF, tablo, görsel ve uzun içerik

Büyük bir dosyayı yüklemek her görevde bütün içeriğinin aynı biçimde işlendiği anlamına gelmez. Ürünler metin çıkarma, görsel işleme veya içerik getirme yöntemlerini farklı kullanabilir. Dosya boyutunu tek başına token sayısı sanmayın; mümkün olduğunda ilgili sayfa/aralığı belirtin.

İçerik	Daha verimli yaklaşım	Korunacak bilgi
Uzun PDF	Bölüm/sayfa seç; yinelenen ekleri ayır	Tablo başlığı, dipnot ve kaynak
Excel / CSV	İlgili sütun ve tarih aralığını seç; yerelde topla	Birim, eksik değer, formül mantığı
Ekran görüntüsü	Yalnız ilgili bölgeyi okunaklı ver	Hata metni ve gerekli çevre bağlamı
Log	İlk ilgili hata ve sınırlı öncesi/sonrası	Sürüm, zaman, korelasyon kimliği
Ses / video	Görevle ilgili zaman aralığı veya doğru transkript	Konuşmacı, zaman damgası, gerekli ses/görsel kanıt

İki aşamalı okuma

Önce içerik haritasını çıkarın: “Karar sorusunu yanıtlamak için hangi bölümler gerekli?” Sonra yalnız seçilen bölümleri ayrıntılı inceletin. Küçük bir belge için fazladan haritalama turu gereksiz olabilir; yöntemi gerçekten büyük kaynaklarda kullanın.

Düz metin anlamı koruyorsa ekran görüntüsü yerine temiz metin göndermek uygun olabilir. Yerleşim, grafik veya görsel durum sorunun parçasıysa görseli koruyun. “Daha düşük çözünürlük her zaman daha ucuzdur” kuralı kullanmayın; işleme ve tokenlaştırma modele bağlıdır. Gemini sayım belgesi metin dışı girdilerin de tokenlaştırıldığını açıklar. [25]

Kaynak talimatı örneği

```
Yalnız sayfa 12-18 ve Tablo 4’ü kullan.
Her sonucu sayfa veya tablo adıyla eşleştir.
Eksik veri varsa belirt; belgenin geri kalanını
incelemek gerekirse önce hangi bölümün neden gerekli olduğunu söyle.
```

> Dışlama bir güvenlik mekanizması değildir. Hassas veriyi paylaşmadan önce gereksiz kişisel alanları yerelde kaldırın. Sırf tasarruf için ilgili dipnotları veya aykırı kayıtları silip sonucu çarpıtmayın.

Kopyalanabilir istem kütüphanesi

Köşeli parantezleri kendi bilgilerinizle değiştirin. Bu şablonlar ürün komutu değildir; üç sağlayıcının sohbetlerinde veya uygun kodlama ajanında görev talimatı olarak uyarlanabilir.

1. Kaynakla sınırlı kısa yanıt

```
Amaç: [karar sorusu].
Kaynak: [dosya/bölüm/veri aralığı].
Yalnız bu kaynaklardan yanıtla; çıkarımı ayrıca etiketle.
Çıktı: [3 madde / küçük tablo], en çok [150] kelime.
Eksik bilgi varsa uydurma; yalnız kritik eksiği belirt.
```

2. Gereksiz yeniden üretimi önleme

```
Önceki çıktının yalnız [bölümünü] değiştir.
Düzeltme: [somut talimat].
Sayılar, kaynaklar ve diğer kararlar aynı kalsın.
Yalnız değişen kısmı ver; tamamını yeniden yazma.
```

3. Dar kapsamlı hata düzeltme

```
Hata: [girdi, beklenen, gerçekleşen].
Başlangıç dosyaları: [yollar].
Kabul: [gözlenebilir davranış + ilgili test].
Önce gerekli dosyaları ve hatanın nedenini doğrula.
En küçük tutarlı düzeltmeyi yap; ilgili testleri çalıştır.
Çıktı: değişiklik, test kanıtı, kalan risk.
İki farklı yaklaşım aynı nedenle başarısızsa
döngüyü durdur; engeli ve gereken kanıtı bildir.
```

4. Kontrollü araştırma

```
Karar: [seçim]. Coğrafya / dönem: [sınır].
Önce en çok [5] ilgili birincil kaynağı incele.
Kararı değiştirecek eksik kanıt varsa aramayı genişlet.
Çıktı: 3 bulgu, belirsizlikler, öneri ve kaynak linkleri.
Aynı bilgiyi tekrarlayan sayfaları özetlemeye devam etme.
```

> Sabit “en çok 5 kaynak” veya “iki deneme” eşikleri başlangıç tercihidir. Araştırmanın karmaşıklığı ve hatanın riski gerektiriyorsa sınırı açıkça yükseltin. Sayfa 6’daki devir özetiyle bu şablonları birlikte kullanabilirsiniz.

Tasarrufu gerçekten ölçme

“Daha az yazdım, daha az harcadım” varsayımını kontrollü bir karşılaştırmayla sınayın. Aynı işi, aynı veri ve aynı kabul ölçütleriyle ölçün. Tek seferlik kolay bir sonuçtan genel bir yüzde çıkarmayın.

Küçük bir A/B deneyi

- 5-10 temsilî görev seçin: kısa yazı, belge sorusu, küçük hata, birkaç dosyalı değişiklik.
- A akışında mevcut yöntemi; B akışında tek bir iyileştirmeyi kullanın. Örneğin yalnız kaynak kapsamını daraltın.
- Kod için aynı başlangıç sürümünü ve temiz çalışma durumunu kullanın; önceki çözümün sonucu kolaylaştırmasını önleyin.
- Model, çaba ve etkin araçlar aynı kalsın. Soğuk/sıcak önbellek koşulunu ayrı kaydedin.
- Başarıyı önceden tanımlayın; toplam tüketim, süre, tekrar ve kaliteyi birlikte karşılaştırın.

Kayıt alanı	Örnek / açıklama
Ürün, istemci, tarih ve model	CLI sürümü; normal / yüksek çaba
Görev ve kabul sonucu	Başarılı / kısmi / başarısız
Girdi / cache / çıktı / düşünme	Yalnız sunulan güvenilir alanlar
Araç çağrısı ve tekrar	Başarısız denemeler dahil
Toplam süre / gerçek maliyet	API paneli; abonelikte görülen kota değişimi
Kalite ve kusur	Test, kaynak doğruluğu, gerekli düzeltme

```
Tasarruf (%) = 100 × (baz tüketim - yeni tüketim) / baz tüketim
Başarı başına tüketim = tüm denemelerin toplamı / kabul edilen iş
Cache payı = cache okuma / toplam girdi [aynı tanım ve API ile]
```

Örneğin 8 görevde 6 kabul edilmiş sonuç aldıysanız, yalnız başarılı altı çağrıyı saymayın; diğer denemelerin tüketimi de paya girer. Abonelik göstergesi yuvarlanmışsa veya token ayrıntısı yoksa sonucu “yaklaşık kota değişimi” diye raporlayın.

> Farklı sağlayıcıların ham token sayısı tek başına kalite veya fiyat sıralaması değildir. Tokenizer, model, cache, araç zinciri ve kabul edilen sonuç farklı olabilir. Karşılaştırmayı aynı tamamlanmış iş üzerinden yapın.

Sorun giderme ve yaygın yanılgılar

Belirti / iddia	Kontrol ve düzeltme
“Kısa sordum, çok tüketti.”	Geçmiş, dosyalar, düşünme ve araç çağrılarını inceleyin.
“Yeni sohbet kotayı sıfırlar.”	Yeni bağlam açılır; tüketilmiş kullanım iade edilmez.
“Cache varsa token yoktur.”	Cache girdi tekrarını daha uygun fiyatlayabilir; bağlam hâlâ vardır.
“Her zaman İngilizce daha ucuz.”	Tokenizer ve görev değişir; Türkçe anlam kaybını da ölçün.
“Her zaman düşük çaba en iyisi.”	Başarısızlık ve yeniden deneme toplamını karşılaştırın.
“Çoklu ajan tasarruf sağlar.”	Süre azalabilir; toplam tüketim artabilir.
“Ctrl+L geçmişi siler.”	Gemini CLI’de görünümü temizler; /clear ayrıdır. [16]
“Tüm depoyu baştan oku.”	Önce dosya/sembol keşfi, sonra ilgili kaynaklar.
“AGENTS.md’ye koyarsam ücretsiz.”	Yüklenen talimatlar bağlam kullanır; kısa ve güncel tutun.
“Aynı komut her yerde geçerli.”	Ürün, CLI/IDE yüzeyi ve sürümü doğrulayın.

Beklenmedik artışta teşhis sırası

- Kimlik doğrulama ve faturalama türü değişti mi? Abonelikten API’ye geçilmiş olabilir.
- Model, çaba düzeyi veya uzun bağlam davranışı değişti mi?
- Yeni eklenti, MCP, global kural veya otomatik görev eklendi mi?
- Test çıktısı ya da büyük dosya yanlışlıkla bağlama girdi mi?
- Önbellek süresi doldu mu veya sabit içerik değişerek yeniden yazım mı tetiklendi?
- Aynı hata etrafında araç ve düzeltme döngüsü oluştu mu?

> “Zincirleme düşünmeni göster” istemek tasarruf yöntemi değildir. Gerekli olanı isteyin: kısa gerekçe, doğrulanabilir kanıt, önemli varsayım ve sonuç. Gizli düşünme miktarını yalnız görünür açıklamanın uzunluğundan çıkarmayın.

7 günlük uygulama ve son kontrol

Gün	Uygulama
1	Mevcut akışı ölçün; 5-10 görev ve başarı ölçütü seçin.
2	Başlangıç istemine hedef, kapsam ve kabul ölçütü ekleyin.
3	Dosya, sayfa ve log kapsamını daraltın; kaliteyi karşılaştırın.
4	Kalıcı talimatları sadeleştirin; tekrar ve eski kuralları çıkarın.
5	Model/çaba seçeneklerini aynı görevlerde deneyin.
6	Araç zincirini ve varsa API cache davranışını ölçün.
7	Başarı başına tüketimi karşılaştırın; işe yarayan ayarları koruyun.

Her işten önce

- Bir teslimat ve gözlenebilir kabul ölçütü var mı?
- Gerekli bağlam erişilebilir ve güncel mi?
- Kullanılan ürün, hesap türü, model ve çaba belli mi?
- Çıktı biçimi, kaynak kapsamı ve bitiş koşulu yazılı mı?

Her işten sonra

- Sonuç doğru mu; gereken test veya kaynak doğrulaması yapıldı mı?
- Toplam deneme ve araç tüketimi dahil edildi mi?
- Yeni işe geçilecekse kısa devir özeti yeterli mi?
- Kalıcı kurala gerçekten yeni madde gerekiyor mu?

Kısa terimler sözlüğü

Bağlam: modelin o istekte kullandığı çalışma içeriği. **Tokenizer:** içeriği token birimlerine ayıran sistem. **Compaction:** eski geçmişi daha küçük bir temsile/özete dönüştürme. **Cache:** tekrar kullanılan içeriğin hesaplama/ücret avantajı için yeniden kullanılması. **MCP:** modellere araç ve kaynak bağlama protokolü. **Çaba:** desteklenen modelin düşünme derinliğini etkileyen ayar.

> Kalıcı hedef: daha az gereksiz bağlam, daha az yanlış deneme ve kabul edilen sonuç başına daha düşük tüketim. Kuralları ayda bir veya ürün güncellemesi sonrası gözden geçirin; sabit bir tasarruf yüzdesi beklemeyin.

Resmî kaynaklar / 1

Numaralar metindeki bağlantılarla eşleşir. Başlıklara tıklayarak resmî sayfalara ulaşabilirsiniz. Tüm bağlantılar 17 Kasım 2026 tarihinde kontrol edilmiştir.

[01] OpenAI | Understanding and counting tokens
help.openai.com

[02] Anthropic | Usage limit best practices
support.claude.com

[03] Claude Code | Manage costs effectively
code.claude.com

[04] Claude Code | How Claude remembers your project
code.claude.com

[05] Claude Code | Extend Claude with skills
code.claude.com

[06] OpenAI | Codex / Work best practices
learn.chatgpt.com

[07] OpenAI | Developer commands (CLI)
learn.chatgpt.com

[08] OpenAI | Codex / Work pricing and usage
learn.chatgpt.com

[09] OpenAI | Custom instructions with AGENTS.md
learn.chatgpt.com

[10] OpenAI | Projects in ChatGPT
help.openai.com

[11] OpenAI | Prompt engineering best practices for ChatGPT
help.openai.com

[12] Google | Gemini Apps limits and feature availability
support.google.com

[13] Google | Tips for creating custom Gems
support.google.com

[14] Google | Gemini Code Assist consumer account deprecation
developers.google.com

[15] Google Cloud | Code Assist Standard and Enterprise sunset
docs.cloud.google.com

[16] Gemini CLI | CLI commands
geminicli.com

Resmî kaynaklar / 2

Numaralar metindeki bağlantılarla eşleşir. Başlıklara tıklayarak resmî sayfalara ulaşabilirsiniz. Tüm bağlantılar 17 Kasım 2026 tarihinde kontrol edilmiştir.

[17] Gemini CLI | Ignoring files (.geminiignore)
geminicli.com

[18] Gemini CLI | Token caching and cost optimization
geminicli.com

[19] Antigravity CLI | Context command
antigravity.google

[20] Antigravity CLI | Effort command
antigravity.google

[21] Antigravity CLI | AI credits command
antigravity.google

[22] Antigravity CLI | Migration from Gemini CLI
antigravity.google

[23] OpenAI API | Prompt caching
developers.openai.com

[24] OpenAI API | Reasoning models
developers.openai.com

[25] Gemini API | Understand and count tokens
ai.google.dev

[26] Gemini API | Context caching
ai.google.dev

[27] Claude API | Token counting
platform.claude.com

[28] Claude API | Prompt caching
platform.claude.com

[29] OpenAI API | Batch API
developers.openai.com

[30] Google Cloud | Code with Gemini Code Assist
docs.cloud.google.com

[31] Gemini CLI | Provide context with GEMINI.md
geminicli.com

> Kaynakların ürün sınırları veya komutları değişebilir. Google geçişinde kaynak [14]-[15] güncel hesap durumunu belirlemek; [16]-[18] ise uygun eski CLI kurulumlarını anlamak içindir. Kaynaklar arasında tarih/sürüm farkı varsa güncel resmî geçiş duyurusu ve kurulu sürümün yardımı önceliklidir.