Claude Code Kurulum ve Token Tasarrufu Rehberi

Özet ve hızlı başlangıç

Token harcamasını en çok düşüren şey model seçimi değil, **sohbetin boyu**: her mesajda geçmişin tamamı yeniden gönderilir. Bu rehber kurulumu, otomatik cache'i bozmamayı, işe göre model seçmeyi ve veri kazımada büyük çıktıyı modele hiç göstermemeyi anlatır.

İlk gün yapılacak 10 şey:

1. Claude Code'u kur, `claude` ile giriş yap (Bölüm: Kurulum).
2. Her projeye ayrı klasör aç, `claude` komutunu o klasörde çalıştır.
3. `~/.claude/CLAUDE.md` dosyasına aşağıdaki tasarruf şablonunu koy; kısa tut (en fazla 1–2 sayfa).
4. Varsayılan modeli **Sonnet** yap; zor planlama için geçici olarak Opus/Fable'a geç.
5. Basit, tekrarlı işleri yapan alt ajanlara `model: haiku` yaz.
6. Bir iş bitince `/clear`; aynı iş uzayınca `/compact`.
7. `/context` ve `/cost` ile neyin yer kapladığını düzenli kontrol et.
8. Kazıma sonucunu dosyaya yazdır; modele ham HTML değil, 20 satırlık örnek ve özet göster.
9. Gerek olmayan MCP sunucularını kapat; her biri başlangıç bağlamına yük bindirir.
10. Sohbet uzayınca `DEVAM.md` yazdırıp yeni sohbet aç.

Kurulum

Kurulum tek komuttur; Node.js gerekmez. Bir Claude Pro/Max aboneliği veya API anahtarı gerekir.

1. **Windows (PowerShell):**

   ```
   irm https://claude.ai/install.ps1 | iex
   ```

   macOS / Linux:

   ```
   curl -fsSL https://claude.ai/install.sh | bash
   ```

2. Windows'ta Git for Windows kurulması önerilir (zorunlu değil): Claude'a Git Bash araçlarını (`grep`, `head`, `sed`) verir, bunlar büyük çıktıyı kırpmak için lazım.
3. Kontrol: `claude --version`
4. Proje klasörüne gir, `claude` yaz, açılan tarayıcıdan giriş yap.
5. Python için: Python 3.11+ kur, her projede sanal ortam aç (`python -m venv .venv`). Claude'a "bu projede `.venv` kullan" demek yeterli.
6. İlk komutlar:
   - `/init` — projeyi tarar, proje `CLAUDE.md` taslağı çıkarır
   - `/model` — model seçer (varsayılan olarak kaydeder)
   - `/context` — bağlamın ne ile dolduğunu renkli tablo olarak gösterir
   - `/usage` (veya `/cost`) — harcama ve plan limiti
   - `/resume` — eski bir sohbete döner

İpucu: VS Code kullanılıyorsa Claude Code eklentisi aynı motoru kullanır; terminal ile aynı ayarları okur.

Token ekonomisi: cache ve chunk

Cache'i kurmaya gerek yok, Claude Code onu **otomatik** yapar; yapılması gereken onu bozmamak ve sohbeti kısa tutmaktır.

Neden sohbet uzadıkça pahalanır

Her mesajda sistem talimatları, CLAUDE.md, araç listesi ve **tüm geçmiş** modele yeniden gönderilir. 50 bin token'la başlayan bir sohbet 350 bine çıktığında aynı "tamam devam et" mesajı yaklaşık 7 kat pahalıya gelir. Cache bu tekrarı ucuzlatır ama sıfırlamaz: cache'ten okunan token normal fiyatın yaklaşık onda biridir.

Cache nasıl çalışır

- Süre: Claude aboneliğinde ana sohbet için 1 saat; API anahtarıyla 5 dakika. Bu süre içinde yazılan her mesaj geçmişi ucuz okur.
- API anahtarıyla çalışılıyorsa 1 saatlik cache ayarla açılabilir: `settings.json` içinde `"env": {"CLAUDE_CODE_PROMPT_CACHE_TTL": "1h"}`.
- **Cache'i sıfırlayanlar:** sohbet ortasında model değiştirmek (`/model`), effort seviyesini değiştirmek, MCP sunucusu eklemek/kaldırmak, plugin açıp kapatmak, `/compact`, Claude Code güncellemesi. Bunları sohbetin başında yap, ortasında yapma.
- Sohbet ortasında CLAUDE.md'yi düzenlemek cache'i bozmaz ama değişiklik `/clear` veya yeniden başlatmaya kadar uygulanmaz.

Bağlamı küçük tutma araçları

Komut / yöntem	Ne zaman
`/clear`	İş bitti, yeni işe geçiliyor. En güçlü tasarruf.
`/compact <odak>`	Aynı iş sürüyor ama geçmiş şişti. Örnek: `/compact sadece kazıma script'inin son hali ve açık hataları tut`
`DEVAM.md` devri	Sohbet ~150k'yı geçti: Claude durumu dosyaya yazar, yeni sohbet onu okuyup devam eder, sonra siler. `/compact`'tan daha temiz.
Alt ajan (subagent)	Geniş arama/okuma işi: alt ajan 50 dosya okur, ana sohbete sadece 20 satırlık sonuç döner.
`/context`	Başlangıç yükünü görmek: boş sohbet 20–30k'nın üstündeyse MCP/skill/CLAUDE.md fazla.

Chunk: dosyayı parça parça okutmak

Claude dosyaları zaten satır aralığıyla okuyabilir; önemli olan onu doğru yönlendirmek:

- "Tümünü oku" değil, "önce `grep` ile `def parse` ara, sadece o fonksiyonu oku".
- Büyük CSV/JSON için: "ilk 20 satırı ve sütun adlarını göster", analizi Python'a yaptır.
- Uzun log için: "son 50 satır" veya "`ERROR` geçen satırlar".
- PDF için: "1–5. sayfaları oku", hepsini birden değil.

Bu kurallar CLAUDE.md şablonunda zaten var; bir kez yazılınca Claude kendisi uygular.

Daha ileri: bağlam sıkıştırma

Araç çıktılarını modele gitmeden sıkıştıran üçüncü taraf araçlar da var (örn. Headroom eklentisi). Temel kurulum oturduktan sonra denenebilir; ilk gün gerekmez.

Model seçimi: her işe en iyisi gerekmez

Varsayılan **Sonnet** olsun; Opus/Fable yalnız zor düşünme için, Haiku basit tekrarlı işler için. Model sohbetin başında seçilmeli, ortasında değiştirmek cache'i sıfırlar.

Model	Ne için	Maliyet
`haiku` (Haiku 5.5)	Dosya arama, basit düzenleme, veri etiketleme, alt ajan işleri	En düşük

Model	Ne için	Maliyet
`sonnet` (Sonnet 5.5)	Günlük kodlama, kazıma script'i yazma, hata ayıklama	Orta
`opus` (Opus 5.5)	Mimari karar, karmaşık hata, çok dosyalı büyük değişiklik	Yüksek
`fable` (Fable 5.1)	En zor problemler; en yeni ve en güçlü model	En yüksek
`opusplan`	Planlama aşamasında Opus, uygulamada Sonnet — otomatik	Dengeli

Otomatik yönlendirme nasıl kurulur

Claude'un işe göre modeli "kendi ayarlaması" üç katmanla olur:

1. **Ana model:** `settings.json` içinde `"model": "opusplan"` veya `"sonnet"`. `opusplan` ile plan modunda (Shift+Tab ile açılır) Opus düşünür, kodu Sonnet yazar. Plan→uygulama geçişinde bir kez cache yenilenir; bu, bütün işi Opus'ta yapmaktan yine ucuzdur. Günlük işlerde düz `sonnet` yeterli.
2. **Alt ajanlar:** `~/.claude/agents/` klasörüne ajan dosyaları koyulur, her birinin başında `model:` yazar. Claude işi uygun ajana devreder, o ajan kendi modeliyle çalışır. Ajana özel `model:` satırı tercih edilmeli; `CLAUDE_CODE_SUBAGENT_MODEL` ortam değişkeni tüm alt ajanları (planlama ajanı dahil) aynı modele bağlar.
3. **Skill'ler:** Skill dosyasının başına `model: haiku` yazılabilir; o skill çalışırken o model kullanılır. Bu da bir model geçişidir ve o tur cache'i bozar; sık çalışan küçük skill'lerde değil, uzun ve bağımsız iş yapanlarda kullan.

Örnek ajan dosyası `~/.claude/agents/kazici.md`:

```
---
name: kazici
description: Bir sitenin HTML yapısını inceleyip seçicileri çıkarmak ve basit
kazıma script'ini denemek için kullan.
model: haiku
---
Görevin: verilen sayfayı indir, cache/ klasörüne kaydet, hedef alanların CSS
seçicilerini bul.
Sonuç olarak sadece seçici listesini ve 3 örnek değeri döndür. Ham HTML
döndürme.
```

Effort (düşünme derinliği)

`/effort low|medium|high|xhigh` ile ayarlanır; yüksek effort daha çok düşünme token'ı harcar. Rutin işlerde `medium`, zor hata ayıklamada `high`. Bunu da sohbetin başında seç: effort değişikliği de cache'i sıfırlar.

CLAUDE.md şablonu

`CLAUDE.md` her oturumun başında (alt ajanlar dahil) otomatik yüklenir: içine yazılan her satır her mesajda tekrar ödenir. Bu yüzden kısa, kural odaklı olmalı. İki seviye var:

- `~/.claude/CLAUDE.md` — tüm projelerde geçerli kişisel kurallar (aşağıdaki şablon).
- `proje-klasoru/CLAUDE.md` — yalnız o projeye özel 5–20 satır (çalıştırma komutu, tuzaklar). Claude'a `/init` dedirterek ilk taslak çıkarılabilir, sonra kısaltılmalı.

Aşağıdaki metni olduğu gibi `~/.claude/CLAUDE.md` dosyasına kopyalayın:

```
# KURALLAR — her projede geçerli

## Çalışma şekli
- Kodu sen yaz, dosyayı sen oluştur, komutu sen çalıştır. Rutin iş için onay sorma.
- Her işi önce fazlara böl, listeyi göster, sonra sırayla yap. Her faz bitince doğrula.
- Çalıştırmadığın kodu "çalışıyor" diye sunma.
- Emin olmadığın kütüphane/API adını uydurma; context7 ile doğrula.

## Token tasarrufu
- Aynı dosyayı iki kez okuma. Bağımsız işleri tek mesajda paralel yap.
- Büyük dosyayı tamamen okuma: önce grep, sonra ilgili satır aralığı.
- Büyük çıktı üreten komutu filtrele (head, tail, grep). Log'un tamamını basma.
- Kazıma/veri işlerinde veriyi dosyaya yaz; bana yalnız satır sayısı + ilk 10 satır göster.
- Sayfanın metni yetiyorsa ekran görüntüsü alma; görüntü sadece görsel sorun için.
- Basit/tekrarlı alt görevleri haiku modelli alt ajana ver.
- Kısa yaz. Uzun açıklama, özet tekrarı yapma.

## Sohbet uzayınca devir
İş parçası bittiğinde veya bağlam ~150k'yı geçtiğinde dur:
1. Proje klasörüne DEVAM.md yaz: nerede kalındı, sıradaki adım, denenip olmayanlar, dosya yolları.
2. "Yeni sohbet aç, DEVAM.md'den devam ederiz" de.
Yeni sohbet DEVAM.md'yi okur, işi yapar, dosyayı siler.
```

```
## Güvenlik
- Her projede ilk iş: .env, anahtarlar, .venv, node_modules, data/ →
.gitignore.
- API anahtarını koda yazma, ortam değişkeninden oku.
- Silme, push --force, reset --hard gibi geri alınamaz işlerden önce sor.

## Hatadan ders
Aynı hatayı iki kez yapma: iş bitince tek satır ders çıkar ("X yaparken Y
oluyor → Z yap")
ve proje README'sinin Tuzaklar bölümüne yaz (en fazla 5 madde).
```

Bu dosyaya **yazılmaması gerekenler:** değişiklik günlüğü, skill/MCP listesi (sistem zaten veriyor), `requirements.txt`'den okunabilen bağımlılıklar. Dosya 25 KB'ı geçerse nadir lazım olan kısımlar ayrı `docs/*.md` dosyasına taşınmalı.

Skills, MCP ve repolar

Skill'ler ucuzdur (kullanılana kadar yalnız adı ve tek satırlık açıklaması yüklenir); MCP sunucuları da araç arama özelliği sayesinde artık yalnız adlarıyla yüklenir. Yine de ihtiyaç olmayanı kurmamak en iyisi.

Skill nedir, nasıl kurulur

Skill, Claude'a "bu işi şöyle yap" diyen hazır talimat paketidir. İki yerde durur:

- `~/.claude/skills/<ad>/SKILL.md` — tüm projelerde
- `proje/.claude/skills/<ad>/SKILL.md` — yalnız o projede

Plugin olarak kurmak (Claude Code içinde yazılır):

```
/plugin marketplace add anthropics/skills
/plugin install document-skills@anthropic-agent-skills
/plugin install superpowers@claude-plugins-official
```

İlki PDF/Word/Excel/PowerPoint okuma-yazma skill'lerini getirir. `superpowers` planlama, test, hata ayıklama gibi çalışma disiplinini ekler. `/plugin` komutu menüden göz atmayı sağlar.

Kendi skill'ini yazmak: tekrar ettiğin bir işi bir kez Claude ile yap, sonra "bu akışı `site-kazima` adında bir skill olarak kaydet" de. Bir sonraki sefer aynı açıklamayı tekrar yazmak gerekmez.

Önerilen MCP sunucuları (az ve öz)

Sunucu	Ne işe yarar	Kurulum (terminalde)
context7	Kütüphanelerin güncel dokümanı; Claude'un eski API uydurmasını önler	`claude mcp add --scope user --transport http context7 https://mcp.context7.com/mcp`
playwright	Tarayıcıyı kontrol eder; JS'li siteleri incelemek, ağ isteklerinde API bulmak	`claude mcp add --scope user playwright -- npx @playwright/mcp@latest`
github	Issue, PR, repo işlemleri	Genelde `gh` komut satırı aracı yeterli; MCP şart değil

`--scope user` tüm projelerde geçerli yapar; yalnız bir projede istenirse bayrak kaldırılır. Kurulu olanlar `claude mcp list` ile görülür, `claude mcp remove <ad>` ile kaldırılır.

Repoları yüklemek

1. Repoyu klonla: `git clone https://github.com/kullanici/repo.git` (ya da Claude'a "bu repoyu klonla" de).
2. Klasöre gir, `claude` çalıştır.
3. İlk mesaj: "Bu repoyu tanı: ne yapıyor, nasıl çalıştırılıyor, kısa özet." Claude bunu alt ajanla yaparsa ana sohbete yalnız özet gelir.
4. `/init` ile proje CLAUDE.md'si oluştur, sonra kısalt.
5. Kendi çalışman için ayrı branch aç: `git switch -c calisma` — ana dal temiz kalır.

GitHub hesabıyla çalışmak için bir kez `gh auth login` (GitHub CLI) yeterli.

Python ile veri kazıma

Alttın kural: **Claude kodu yazar, veriyi Python işler.** Claude'a 2 MB'lık HTML okutmak tek seferde 500 bin token'a kadar yakar; aynı işi yapan script'i yazdırmak birkaç bin token'dır.

Klasör yapısı

```
kazima-projesi/
  CLAUDE.md          # bu projeye özel 5-10 satır kural
  .gitignore         # .env, .venv, data/, cache/
  .env               # API anahtarları, proxy
  requirements.txt
  scrapers/          # site başına bir dosya
  cache/             # indirilen ham HTML (tekrar indirmemek için)
  data/              # temiz çıktı: .csv / .parquet / .jsonl
  logs/
```

Hangi kütüphane ne zaman

Durum	Araç
Statik sayfa, basit HTML	`httpx` veya `requests` + `selectolax` / `BeautifulSoup`
Sayfa JavaScript ile yükleniyor	`playwright` (Python)
Sitenin arka planında JSON API var	Doğrudan API'ye istek — en hızlı ve en temiz yol
Binlerce sayfa, tarama	`scrapy`
Tablo / temizlik / dışa aktarma	`pandas`, `polars`
Tekrar tekrar indirmeyi önleme	`requests-cache` veya `hishel` (httpx için)

Token harcamadan kazıma akışı

1. **Keşif:** Claude'a sitenin adresini ver, "bir sayfayı indir, HTML'i `cache/` klasörüne kaydet, bana sadece hedef alanların CSS seçicilerini çıkar" de. Tarıyıcı aracı varsa ağ isteklerinde gizli JSON API arat.
2. **Script:** Script'i yazsın, 3 sayfada denesin, sonucu dosyaya yazsın.
3. **Doğrulama:** Ekrana sadece `head -20` ve satır sayısı basılsın, tamamı değil.
4. **Büyük çalıştırma:** Uzun süren işi arka planda çalıştırsın; ilerleme `logs/` dosyasına gitsin, Claude yalnız son satırları okusun.
5. **Hata:** Patlayan sayfanın sadece URL'si ve hata mesajı gösterilsin.

Chunk (parça parça işleme)

- Liste sayfalarını sayfa sayfa işle, her 100 kayıtta dosyaya ekle (`.jsonl` en kolayı). Yarıda kesilirse kaldığı yerden devam etsin.

- Büyük bir dosyayı Claude'a okutman gerekirse "ilk 200 satırını oku" veya "`price` geçen satırları grep'le" de; Claude dosyayı satır aralığı ile okuyabilir.
- Metni yapay zeka ile işlemek gerekiyorsa (özetleme, sınıflandırma), bunu Claude Code sohbeti üzerinden değil, script içinden API ile ve ucuz modelle yap: bkz. "Diğer yapay zekalar".

Nezaket ve hukuk

Sitenin kullanım şartlarına ve `robots.txt` dosyasına bak, istekler arasına gecikme koy (1–3 sn), kişisel veri topluyorsan KVKK yükümlülüklerini göz önünde tut. Giriş gerektiren sayfalarda çerez ve şifreleri yalnız `.env` dosyasında tut.

Diğer yapay zekalar

Claude Code kendi içinde yalnızca Claude modellerini kullanır; başka yapay zekaları **araç olarak** çağırır. İki yol var: Claude'un yazdığı Python script'i bir API'yi çağırır, ya da bir MCP sunucusu o servisi Claude'a araç olarak ekler.

İş	Önerilen yol	Not
Binlerce metni etiketleme / özetleme	Script içinden Claude Haiku API veya başka ucuz bir model	Sohbet üzerinden yapılırsa her satır bağlama girer
Toplu iş, acil değil	Anthropic **Batch API**	Normal fiyatın yaklaşık yarısı, sonuç saatler içinde
Görsel üretme	Google Gemini / OpenAI görsel API'si veya bir görsel MCP sunucusu	Claude görseli okur ve tasarlar ama piksel üretmez
PDF, Word, Excel, PowerPoint	Claude'un resmi belge skill'leri (pdf, docx, xlsx, pptx)	Kurulum: bkz. Skills bölümü
Ses → yazı	Whisper (yerel veya API)	Script ile
Tamamen ücretsiz / yerel	Ollama ile yerel model	Basit sınıflandırma için yeterli, kodlama için zayıf
Tek anahtarla çok model	OpenRouter	Farklı modelleri aynı kodla denemek için

Pratik kural: API anahtarlarını `.env` dosyasına koy, `.gitignore`'a ekle, Claude'a "anahtarı koda yazma, `os.getenv` ile oku" de. CLAUDE.md'ye tek satır ekle: "Toplu metin işleme ve görsel üretme işlerini `tools/llm.py` üzerinden yap" — Claude bu yardımcı dosyayı bir kez yazar, sonra hep onu kullanır ve işe göre modeli kendisi seçer.

Ayar dosyası: settings.json

Aşağıdaki dosya `~/.claude/settings.json` olarak kaydedilince tüm projelerde geçerli olur; proje klasöründeki `.claude/settings.json` bunu ezer.

```
{
  "model": "sonnet",
  "effortLevel": "medium",
  "language": "Turkish",
  "autoCompactEnabled": true,
  "permissions": {
    "allow": [
      "Bash(python:*)",
      "Bash(pip install:*)",
      "Bash(pip list:*)",
      "Bash(git status:*)",
      "Bash(git diff:*)",
      "Bash(git log:*)",
      "Bash(ls:*)",
      "Bash(head:*)",
      "Bash(tail:*)",
      "Bash(grep:*)",
      "Bash(wc:*)"
    ],
    "deny": [
      "Bash(rm -rf:*)",
      "Bash(git push --force:*)",
      "Bash(git reset --hard:*)",
      "Read(./.env)"
    ]
  }
}
```

Ne yapar:

- `model` / `effortLevel` — varsayılan model ve düşünme derinliği (bkz. Model seçimi).
- `language` — Claude her zaman Türkçe yanıt verir.
- `allow` — bu komutlar her seferinde onay istemez; zararsız okuma komutları buraya.
- `deny` — geri alınamaz komutlar ve `.env` okuması tamamen engellenir.
- API anahtarıyla çalışılıyorsa `env` içine `"CLAUDE_CODE_PROMPT_CACHE_TTL": "1h"` eklenebilir (abonelikte zaten 1 saat).

Durum satırı (statusline)

En faydalı gösterge: o an hangi model ve bağlamın yüzde kaçı dolu. Claude Code içinde şu yazılır:

```
/statusline model adını, bağlam doluluk yüzdesini ve bu oturumun maliyetini göster
```

Claude script'i kendisi yazar ve ayara ekler. Bağlam ~150k token'ı geçince `DEVAM.md` + yeni sohbet zamanı gelmiştir.

İleri seviye: hook ile kural hatırlatma

Hook, belirli bir olayda (örn. dosya yazmadan önce) otomatik çalışan küçük bir script'tir. Örneğin CLAUDE.md'ye yazılmadan önce dosya boyutunu kontrol edip 25 KB üstündeyse Claude'u uyaran bir hook, dosyanın şişmesini kendiliğinden engeller. Claude'a "CLAUDE.md'ye yazmadan önce boyutunu kontrol eden bir PreToolUse hook'u kur" demek yeterli.

Günlük alışkanlıklar ve kontrol listesi

İyi kurulum token'ın yarısını kurtarır; diğer yarısını alışkanlıklar kurtarır.

Yap:

- Bir iş = bir sohbet. Kazıma bitti, rapor yazdıracaksın → `/clear`.
- İsteği ilk mesajda net yaz: hedef site, istenen alanlar, çıktı biçimi, örnek satır. Belirsiz istek = gidip gelen 10 mesaj.
- Dosyayı gösterirken yol ver (`@data/urunler.csv` gibi), içeriği yapıştırma.
- Hata altında tam log yerine son 30 satırı ver.
- Aynı işi tekrar tekrar yaptırıyorsan onu skill'e çevir: "bu adımları bir skill olarak kaydet".
- Uzun bekleme sonrası (öğle arası) devam edeceksen: cache süresi dolmuş olabilir; yeni sohbete `DEVAM.md` ile başlamak daha ucuz olabilir.

Yapma:

- Sohbet ortasında modelçi değiştirip geri dönme: her değişiklik cache'i sıfırlar, geçmişin tamamı tam fiyattan yeniden okunur.
- "Kontrol için" aynı sayfanın ikinci ekran görüntüsünü isteme (her görüntü 1.500–2.500 token ve sohbette kalır).
- 20 MCP sunucusu, 50 skill yükleyip unutma — kullanmadığını kapat.
- Opus/Fable'ı "README düzelt", "dosyayı yeniden adlandır" gibi işlerde kullanma.

Haftalık kontrol:

- [ ] `/context` — başlangıç bağlamı ne kadar? (boş sohbette 20–30k üstü ise MCP/skill/CLAUDE.md fazla)
- [ ] `~/.claude/CLAUDE.md` boyutu 25 KB altında mı?
- [ ] Kullanılmayan MCP sunucuları kaldırıldı mı?
- [ ] `.gitignore` her projede `.env` ve `data/` içeriyor mu?

Kaynaklar

- Kurulum
- Komutlar
- Model ayarları
- Prompt caching
- Alt ajanlar
- Skills
- Plugin kurulumu
- MCP
- Ayarlar referansı