.claude/ Klasörünün Anatomisi

CLAUDE.md, Komutlar, Kurallar, Beceriler,
Ajanlar ve İzin Yapılandırması — Eksiksiz Rehber

Dr. Murat Altun | Claude Code Workshop

Neden .claude/ Klasörü Önemli?

Claude Code'un proje davranışını kontrol eden merkez.

Talimatlar	CLAUDE.md ile Claude'un projedeki tüm davranışını tanımla
Özel Komutlar	Ekip genelinde paylaşılan slash komutları oluştur
Beceriler	Claude'un otomatik tetiklediği iş akışları tanımla
Ajanlar	Uzman alt-ajan kişilikleri yapılandır
İzinler	Hangi araçların kullanılabileceğini kontrol et

İki Klasör, Tek Sistem

Proje düzeyinde ekip yapılandırması + Global kişisel tercihler

proje/.claude/

Ekip yapılandırması

Git'e commit edilir

Herkes aynı kuralları alır

İçerik:

CLAUDE.md, settings.json

commands/, rules/

skills/, agents/

~/.claude/

Kişisel tercihler

Makine-yerel durum

Tüm projelerde geçerli

İçerik:

CLAUDE.md (global)

commands/, skills/, agents/

projects/ (oturum geçmişi)

CLAUDE.md

Claude'un Talimat Kılavuzu

CLAUDE.md Nasıl Çalışır?

1. Claude Code oturumu başladığında ilk okunan dosya
2. Sistem promptuna doğrudan yüklenir — tüm oturum boyunca aktif
3. Proje kökünde → ekip geneli, alt dizinlerde → klasöre özel
4. ~/.claude/CLAUDE.md → tüm projelerde geçerli global tercihler
5. Birden fazla CLAUDE.md varsa hepsi birleştirilir

CLAUDE.md'ye ne yazarsanız, Claude onu takip eder.

CLAUDE.md'de Ne Olmalı?

Yaz

- Build, test, lint komutları
- Mimari kararlar (monorepo, ORM vb.)
- Beklenmeyen tuzaklar ve gotcha'lar
- Import kuralları, adlandırma kalıpları
- Hata yönetimi stilleri
- Dosya/klasör yapısı açıklaması

Yazma

- Linter/formatter config'i (zaten var)
- Link verilebilecek tam dökümantasyon
- Uzun teorik açıklamalar

200 satırın altında tut! Daha uzun dosyalar bağlamı yer,
talimat uyumu düşer.

Etkili Bir CLAUDE.md Örneği

~20 satır — Claude'un üretken çalışması için yeterli

```
# Project: Acme API

## Commands
npm run dev          # Dev server
npm run test         # Jest testleri
npm run lint         # ESLint + Prettier
npm run build        # Production build

## Architecture
- Express REST API, Node 20
- PostgreSQL via Prisma ORM
- Handlers: src/handlers/

## Conventions
- Zod ile request validation
- Dönüş: her zaman { data, error }
- Stack trace client'a gösterilmez
- console.log yerine logger modülü

## Watch out for
```

CLAUDE.local.md

Kişisel tercihler — ekiple paylaşılmaz

Kişiye Özel Geçersiz Kılma

Proje kökünde oluşturulur. CLAUDE.md ile birlikte okunur ama otomatik olarak .gitignore'a eklenir. Kişisel test runner tercihi, dosya açma kalıbı gibi bireysel ayarlar için idealdir.

Nasıl Çalışır?

Claude tüm CLAUDE.md + CLAUDE.local.md dosyalarını birleştirir. Local dosya, ana dosyadaki kuralları geçersiz kılabilir veya ekleyebilir. Repo'ya hiçbir zaman gitmez.

Ekip kuralları CLAUDE.md'de, bireysel tercihler CLAUDE.local.md'de.

rules/ Klasörü

Modüler ve Ölçeklenebilir Talimatlar

rules/ ile Modüler Talimatlar

CLAUDE.md büyüdüğünde kuralları konuya göre ayır

```
.claude/rules/
├── code-style.md
├── testing.md
├── api-conventions.md
└── security.md
```

Odaklı	Her dosya tek bir konuya ayrılmış
Bağımsız	Ekip üyeleri kendi alanını düzenler
Otomatik	CLAUDE.md ile birlikte yüklenir

Path-Scoped Kurallar

YAML frontmatter ile kural dosyasını belirli dosya yollarına bağla:

```
---
paths:
  - "src/api/**/*.ts"
  - "src/handlers/**/*.ts"
---
# API Design Rules
- Tüm handler'lar { data, error } döner
```

commands/ Klasörü

Özel Slash Komutları

Özel Komutlar Oluşturmak

Dosya adı = Komut adı: review.md → /project:review

```
---
description: Birleştirme öncesi diff kontrolü
---
## Değişiklikler

!`git diff --name-only main...HEAD`

## Detaylı Diff

!`git diff main...HEAD`

Yukarıdaki değişiklikleri incele:
1. Kod kalitesi sorunları
2. Güvenlik açıkları
3. Eksik test kapsamı
4. Performans endişeleri
```

Önemli:

!` ` sözdizimi shell komutlarını çalıştırıp çıktıyı prompt'a gömer.

Sadece kayıtlı metin değil, gerçek veri enjeksiyonu.

Konum:

`.claude/commands/`
→ /project:komut

`~/.claude/commands/`
→ /user:komut

Komutlara Argüman Geçirme

$ARGUMENTS ile komut adından sonra gelen metin yakalanır:

```
---
description: GitHub issue'yu araştır ve düzelt
argument-hint: [issue-numarası]
---
#$ARGUMENTS numaralı issue'ya bak.

!`gh issue view $ARGUMENTS`

Bug'ı anla, kök nedeni bul, düzelt
ve yakalayacak bir test yaz.
```

Kullanım:

`/project:fix-issue 234` → Issue #234'ün içeriği otomatik olarak prompt'a eklenir

skills/ Klasörü

Otomatik Tetiklenen İş Akışları

Skills: Otomatik Tetikleme

Komutlar sizi bekler. Beceriler konuşmayı izler ve doğru anda devreye girer.

```
.claude/skills/
├── security-review/
│   ├── SKILL.md
│   └── DETAILED_GUIDE.md
└── deploy/
    ├── SKILL.md
    └── templates/
        └── release-notes.md
```

```
---
name: security-review
description: Güvenlik denetimi.
  Güvenlik incelemesi veya
  deployment öncesi kullan.
allowed-tools: Read, Grep, Glob
---
Güvenlik açıklarını için analiz yap
@DETAILED_GUIDE.md referans
```

Paket Yapısı

Skills, destek dosyalarını yanına alabilir (@referans ile)

Akıllı Tetikleme

description alanına göre bağlam eşleştiğinde otomatik çalışır

Skills vs Commands

Özellik	Commands	Skills
Tetikleme	Manuel — slash komutu	Otomatik — bağlam eşleşmesi
Yapı	Tek .md dosyası	Klasör + SKILL.md + ek dosyalar
Argüman	$ARGUMENTS ile destekler	Desteklemez
Dosya Referansı	Yalnızca shell çıktısı	@dosya ile yan belgelere erişir
Araç Kısıtlama	Yok	allowed-tools ile kısıtlanır
Kullanım	/project:komut veya /user:komut	Otomatik veya /skill-adı

Commands = kullanıcı tetikler. Skills = Claude tetikler. İkisi de ~/.claude/'da kişisel olabilir.

agents/ Klasörü

Uzman Alt-Ajan Kişilikleri

Subagent Tanımlama

Karmaşık görevler için ayrı bağlam penceresinde çalışan uzman ajanlar

```
---
name: code-reviewer
description: Uzman kod inceleyici. PR
  incelerken veya implementasyonları
  doğrularken PROAKTİF kullan.
model: sonnet
tools: Read, Grep, Glob
---
Kıdemli bir kod inceleyicisisin.

İnceleme yaparken:
- Stil değil, bug'ları işaretle
- Belirsiz değil, spesifik düzeltmeler öner
- Uç durumları ve hata yönetimini kontrol et
- Performans: sadece ölçekte önemliyse belirt
```

Araç Kısıtlama

tools alanı ile ajanın ne yapabileceğini sınırla. Salt okunur görevler için Write verme.

Model Seçimi

model alanı ile ucuz/hızlı model kullan. Haiku okuma işlerinde yeterli, Opus karmaşık iş için.

İzole Bağlam

Ajan kendi penceresinde çalışır, bulgularını sıkıştırıp ana oturuma raporlar.

Kişisel ajanlar: ~/.claude/agents/ — tüm projelerde kullanılabilir

settings.json

İzinler ve Proje Yapılandırması

İzin Sistemi: allow / deny

```
{
  "$schema": "https://json.schemastore.org/
    claude-code-settings.json",
  "permissions": {
    "allow": [
      "Bash(npm run *)",
      "Bash(git status)",
      "Bash(git diff *)",
      "Read", "Write", "Edit"
    ],
    "deny": [
      "Bash(rm -rf *)",
      "Bash(curl *)",
      "Read(./.env)",
      "Read(./.env.*)"
    ]
```

allow listesi

Onay sormadan çalıştırılır:
npm/make scriptleri, git komutları,
dosya okuma/yazma işlemleri

deny listesi

Tamamen engellenir:
rm -rf, curl gibi yıkıcı komutlar,
.env gibi hassas dosya erişimi

Listede olmayan

Claude çalıştırmadan önce
kullanıcıdan onay ister —
güvenli orta yol

settings.local.json → Kişisel izin ayarları (otomatik .gitignore). $schema ile VS Code otomatik tamamlama.

Global ~/.claude/ Klasörü

Tüm projelerde geçerli kişisel alan

CLAUDE.md	Global talimatlar — tüm oturumlara yüklenir. Kişisel kodlama ilkeleri, tercih edilen stiller.
settings.json	Global izin ayarları — tüm projelerde geçerli varsayılan allow/deny kuralları.
commands/	Kişisel komutlar → /user:komut-adı olarak tüm projelerde erişilebilir.
skills/	Kişisel beceriler — proje bağımsız otomatik iş akışları.
agents/	Kişisel ajanlar — favori alt-ajan tanımlarınız.
projects/	Oturum geçmişi ve otomatik hafıza. Claude'un keşfettiği kalıplar burada saklanır. /memory ile yönet.

Tam Dizin Yapısı

```
your-project/
├── CLAUDE.md           # Ekip talimatları
├── CLAUDE.local.md     # Kişisel (gitignore)
└── .claude/
    ├── settings.json       # İzinler
    ├── settings.local.json # Kişisel izinler
    ├── commands/           # Slash komutları
    │   ├── review.md
    │   └── fix-issue.md
    ├── rules/              # Modüler kurallar
    │   ├── code-style.md
    │   └── testing.md
    ├── skills/             # Oto-tetikleme
    │   └── security-review/
    │       └── SKILL.md
    └── agents/             # Alt-ajanlar
        └── code-reviewer.md
```

```
~/.claude/
├── CLAUDE.md        # Global talimatlar
├── settings.json    # Global ayarlar
│
├── commands/        # Kişisel komutlar
│   └── standup.md   # → /user:standup
│
├── skills/          # Kişisel beceriler
│   └── ...
│
├── agents/          # Kişisel ajanlar
│   └── ...
│
└── projects/        # Oturum geçmişi
    └── ...          # + otomatik hafıza
```

Pratik Başlangıç Adımları

1	**/init**	Claude Code'da /init çalıştır — otomatik CLAUDE.md oluşturur. Gereksizleri kırp.
2	**settings.json**	allow/deny kurallarını yaz. En azından run komutlarını izin ver, .env'yi engelle.
3	**İlk Komutlar**	En sık yaptığın 1-2 iş akışı için .claude/commands/ altına komut yaz.
4	**rules/ Bölme**	CLAUDE.md kalabalıklaştığında kuralları .claude/rules/ dosyalarına ayır.
5	**Global CLAUDE.md**	~/.claude/CLAUDE.md'ye kişisel kodlama ilkelerini ekle.

Temel İlke

CLAUDE.md en yüksek getirili dosyanızdır.

Önce onu doğru yapın. Geri kalan her şey optimizasyon.

Küçük başla, ilerledikçe geliştir, altyapı gibi davran.

Dr. Murat Altun