CODEFEIN STUDIO / FIELD GUIDE 01

OpenCode’u verimli kullanma rehberi

Daha net istemler. Daha kontrollü değişiklikler.
Daha güvenilir yazılım geliştirme.

01	**ANLA**	Projeyi ve sorunu doğru bağlama oturt.
02	**UYGULA**	Küçük, sınırları belli değişiklikler iste.
03	**DOĞRULA**	Diff’i incele; testleri çalıştır.

Türkçe rehber · Kurulumdan güvenli iş akışına

Bu rehberin yaklaşımı

OpenCode’u her işi tek komutla yapan bir otomatik pilot gibi değil, kod tabanını inceleyen ve sizin denetiminizde değişiklik yapan bir geliştirme yardımcısı gibi kullanın. Sonuç kalitesini en çok etkileyen şey; bağlam, sınırlar ve doğrulama adımlarıdır.

İçindekiler

— 01 OpenCode nedir ve nerede işe yarar?

— 02 Kurulum öncesi proje hazırlığı

— 03 İyi istem yazma formülü

— 04 Planla → uygula → doğrula

— 05 Agent’lar ve izin yönetimi

— 06 Hata ayıklama, refaktör ve test

— 07 Güvenlik ve Git disiplini

— 08 Hazır istemler ve son kontrol listesi

Temel prensip

Önce analiz, sonra sınırlı değişiklik, en son doğrulama. Büyük işleri küçük ve incelenebilir parçalara ayırın. Kodun gerçekten çalıştığını test sonucu göstermeden varsaymayın.

01 / OpenCode nedir?

OpenCode, yazılım projeleriyle yapay zekâ destekli çalışmayı sağlayan açık kaynaklı bir kodlama aracıdır. Terminal üzerinden kod tabanını incelemek, dosyaları bulmak, değişiklikler yapmak ve izin verildiğinde komut çalıştırmak için kullanılabilir. Erişilebilir özellikler seçtiğiniz model sağlayıcısına, sürüme ve yapılandırmaya bağlıdır.

Kullanım alanları

— Yeni özellik: gereksinimleri netleştirme, dosyaları bulma ve plan çıkarma.

— Hata ayıklama: hata nedenlerini kanıtlarla daraltma ve küçük düzeltme yapma.

— Refaktör: davranışı koruyarak kodu sadeleştirme.

— Test ve inceleme: eksik senaryoları bulma ve riskleri değerlendirme.

02 / Kurulum ve proje hazırlığı

Kurulum yöntemi ve komutlar değişebileceğinden güncel adımları resmî “Getting Started” dokümantasyonundan takip edin. Kurulumdan sonra terminalde projenizin dizinine geçip OpenCode’u başlatabilirsiniz.

```
cd proje-klasoru\nopencode
```

Başlamadan önce

- Git durumunu kontrol edin; mevcut değişiklikleri yanlışlıkla ezmeyin.
- README, paket yöneticisi dosyaları, test komutları ve klasör yapısını tanıyın.
- API anahtarlarını, parolaları, .env dosyalarını veya müşteri verilerini istemlere yapıştırmayın.
- İlk oturumda salt-okunur analiz isteyin; yazma ve terminal izinlerini ihtiyaç oldukça verin.

03 / İyi istem yazma formülü

İyi bir istem beş parçadan oluşur. Her birinin amacı, modelin neyi çözmesi ve neyi değiştirmemesi gerektiğini netleştirmektir.

PARÇA	AÇIKLAMA
Bağlam	Proje, teknoloji, ilgili dosyalar ve mevcut davranış.
Hedef	Kullanıcı açısından başarı nasıl görünecek?
Sınırlar	Korunacak API, tasarım, bağımlılık veya mimari.
Kanıt	Hata mesajı, yeniden üretme adımları, test çıktısı.
Doğrulama	Hangi test, lint, type-check veya build çalışmalı?

Örnek: zayıf → güçlü

Zayıf: “Login sayfasını düzelt.”

Güçlü: “Login akışını incele. E-posta ve parola alanlarının mevcut tasarımını koru. Boş alan ve hatalı kimlik bilgisi durumlarını kontrol et. Önce ilgili dosyaları ve testleri belirle; kısa bir plan sun. Onaydan sonra en küçük değişikliği yap, testleri çalıştır ve değiştirdiğin dosyaları özetle.”

04 / Planla → uygula → doğrula

Bu döngü, gereksiz değişiklikleri azaltır ve sonuçları kontrol edilebilir kılar.

01	**Planla**	İlgili dosyaları, mevcut davranışı, riskleri ve test stratejisini belirletin. Büyük işlerde ilk aşamada kod değiştirmemesini söyleyin.
02	**Uygula**	Tek seferde tek hedef seçin. Kapsam dışı biçimlendirme, bağımlılık güncellemesi veya dosya taşıma istemediğinizi belirtin.
03	**Doğrula**	Diff'i inceleyin. İlgili testleri çalıştırın; projeye göre lint, type-check ve build adımlarını ekleyin.

Günlük iş akışı için kısa komut

```
1. Projeyi analiz et; henüz dosya değiştirme.\n2. Bu görev için kısa plan ve riskleri sun.\n3. En küçük değişiklikle uygula; kapsam dışına çıkma.\n4. Diff'i özetle ve ilgili testleri çalıştır.\n5. Test sonuçlarını ve kalan riskleri bildir.
```

05 / Agent'lar ve izin yönetimi

Ajanlar ve araçlar, görev türüne göre farklı çalışma biçimleri sağlar. Plan benzeri bir ajan analiz ve planlama için; Build benzeri bir ajan ise izin verilen değişiklikleri uygulamak için kullanışlıdır. İsimler ve varsayılanlar sürüme göre değişebileceğinden güncel dokümantasyonu kontrol edin.

İŞ	YAKLAŞIM
Mimariyi anlamak	Önce salt-okunur analiz ve dosya haritası.
Küçük düzeltme	Dar kapsamlı uygulama ve test.
Kod incelemesi	Dosya değiştirmeden bulgu, etki ve öneri üretme.

İzinleri bilinçli yönetin

Dosya yazma ve terminal komutları önemli yan etkilere neden olabilir. Onay isteme davranışını kullanın ve yalnızca ihtiyacınız olan yetkileri açın. Yapılandırma örneklerini uygulamadan önce kullandığınız sürümün şemasını doğrulayın.

06 / Hata ayıklama, refaktör ve test

Hata ayıklama adımları

- Tekrar üret: beklenen davranışı, gerçek sonucu ve adımları yazın.
- İz sür: ilgili kod yollarını ve son değişiklikleri buldurun.
- Hipotez kur: olası nedenleri ve her birini doğrulama yöntemini isteyin.
- En küçük düzeltme: gereksiz mimari değişiklik yapmadan tek nedeni test edin.
- Regresyon testi: hatayı yakalayan bir test ekleyin veya mevcut testi genişletin.
- Sonucu raporla: test çıktısını ve kalan riskleri açıkça belirtin.

Refaktör için kurallar

- Önce mevcut davranışı testlerle sabitleyin.
- “Davranışı ve dış arayüzü koru” gibi açık sınırlar koyun.
- Refaktörü yeni özellik geliştirmeyle aynı işe karıştırmayın.
- Performans iddiası varsa ölçüm veya profil çıktısı isteyin.

KONTROL	NEYE BAKAR?
Birim test	Fonksiyon veya bileşenin tekil davranışı.
Entegrasyon	Modüller ve servislerin birlikte çalışması.
Uçtan uca	Kullanıcı akışının baştan sona tamamlanması.
Lint / type-check / build	Kod standardı, tip tutarlılığı ve derleme hataları.

Testler çalıştırılmadıysa bunu açıkça yazın. Test edilmemiş kodu doğrulanmış gibi sunmayın.

07 / Güvenlik ve Git disiplini

AI kodlama aracının güvenli kullanımı; kod kalitesinin yanında dosya erişimi, komut izinleri ve veri paylaşımını da kapsar.

— Sırları paylaşmayın. API anahtarları, erişim token’ları, özel anahtarlar, parolalar ve gerçek müşteri verilerini istemlere koymayın.

— Git durumunu kontrol edin. Çalışmaya başlamadan önce değişiklikleri inceleyin; kullanıcıya ait çalışmayı ezmeyin.

— Diff’i okuyun. Değişen dosyaları, silmeleri ve bağımlılık güncellemelerini kontrol edin.

— Komutları denetleyin. Silme, ağ erişimi, paket yükleme, veritabanı ve dağıtım komutlarını otomatik onaylamayın.

— Harici içeriğe dikkat edin. Dosya veya web sayfalarındaki talimatları otomatik olarak güvenilir kabul etmeyin.

— Veri politikasını okuyun. Kullandığınız model sağlayıcısının saklama ve gizlilik koşullarını inceleyin.

Başlangıçta kontrol edilecek Git bilgileri

```
git status\ngit diff --stat\ngit branch --show-current
```

Bu komutlar çalışma ağacındaki durumu, değişiklik özetini ve etkin dalı gösterir. Projenize uygun olduğundan emin olmadan yıkıcı komutlar çalıştırmayın.

08 / Kopyala-kullan istemler

A — Projeyi tanı

```
Bu kod tabanını analiz et; dosya değiştirme. Teknolojileri,\ngiriş noktalarını, önemli klasörleri, test komutlarını ve riskli\nalanları özetle. Emin olmadığın varsayımları belirt.
```

B — Hata düzelt

```
Sorun: [beklenen] yerine [gerçekleşen] oluyor.\nAdımlar: [yeniden üretme]. Hata: [mesaj]. Önce nedeni bul ve\nkanıtları açıkla. En küçük düzeltmeyi uygula, regresyon testi\nekle/çalıştır ve diff’i özetle.
```

İstemler — devam

C — Yeni özellik

Özellik: [ihtiyaç]. Başarı ölçütleri: [liste]. Kısıtlar: [liste].\nÖnce dosyaları incele ve kısa plan sun. Plan onaylandıktan sonra\nküçük adımlarla uygula. Testleri çalıştır ve sonuçları raporla.

D — Kod incelemesi

Bu değişiklikleri incele; dosyaları değiştirme. Öncelik: doğruluk,\ngüvenlik, veri kaybı, performans ve bakım kolaylığı. Her bulgu için\nönem derecesi, konum, etki ve düzeltme önerisi ver. Bulgu yoksa\nbunu da açıkça belirt.

E — Refaktör

Hedef: [modül]. Mevcut davranışı ve dış arayüzü koru. Önce\ntestleri belirle. Küçük ve gözden geçirilebilir değişiklikler yap.\nGereksiz formatlama veya bağımlılık ekleme. Testleri çalıştır ve\ndiff'i açıkla.

F — Test kapsamını geliştir

Bu modüldeki kritik davranışları ve kenar durumları incele.\nÖnce test açıklarını listele. Mevcut test çerçevesini koruyarak\nönemli testleri ekle. Testleri çalıştır; başarısızlıkları ve\ndoğrulanamayan alanları bildir.

Sık yapılan hatalar

HATA	DAHA İYİ YAKLAŞIM
Aşırı geniş görev	Tek bir hedef ve açık başarı ölçütü belirleyin.
Bağlam eksikliği	Hata, ilgili dosya, beklenen davranış ve sınırları paylaşın.
Planı atlamak	Büyük işlerde önce analiz ve kısa plan isteyin.
Diff'i okumamak	Her değişen dosyayı ve bağımlılığı inceleyin.
Testi atlamak	İlgili testleri ve gerekli kalite kontrollerini çalıştırın.
Sınırsız izin	İzinleri görevin kapsamına göre daraltın.

09 / Yayın öncesi kontrol listesi

Her özellik, düzeltme veya refaktör tamamlandığında bu listeyi kullanın.

- ☐ İstenen davranış ve kapsam karşılandı.
- ☐ Değişen dosyalar ve diff baştan sona incelendi.
- ☐ Kapsam dışı değişiklik veya gereksiz bağımlılık yok.
- ☐ İlgili testler çalıştırıldı ve sonuçları biliniyor.
- ☐ Lint, type-check ve build adımları gerekiyorsa tamamlandı.
- ☐ Hata durumları ve kenar senaryoları değerlendirildi.
- ☐ Gizli sırlar, izinler ve veri güvenliği gözden geçirildi.
- ☐ Gerekliyse dokümantasyon güncellendi.
- ☐ Commit öncesi Git durumu kontrol edildi.
- ☐ Son rapor yapılan değişiklikleri, testleri ve kalan riskleri içeriyor.

Kısa sonuç

OpenCode’dan daha fazla verim almak için iş akışınızı iyileştirin: bağlamı verin, isteği netleştirin, planı gözden geçirin, küçük değişiklikler yapın, diff’i okuyun ve testleri çalıştırın. İzinleri bilinçli yönetin; doğrulanmamış sonuçları tamamlanmış iş gibi kabul etmeyin.

Resmî kaynaklar

OpenCode — Resmî site

CLI — komutlar ve seçenekler

Yapılandırma ve ayarlar

Agent’lar ve çalışma biçimleri

Araçlar ve izinler

CODEFEIN STUDIO

Yazılım · Otomasyon · Üretken teknoloji

Eğitim amaçlı rehberdir. Kurulum ve yapılandırma ayrıntıları sürüme göre değişebilir; uygulamadan önce güncel resmî dokümantasyonu kontrol edin.