

Hangi Git akışı? Gitflow, GitHub Flow ve trunk-based

Gitflow, GitHub Flow ve trunk-based development arasında ürünün şekline, ekip büyüklüğüne ve deploy temposuna göre seçim yapmak için üç soru ve bir karar matrisi.

01 GİT · WORKFLOW

Hangi Git akışı? Üç soruyla karar

Gitflow da GitHub Flow da doğru; farklı ürün şekilleri için. Trunk-based ise bir tempo kararı. Cevap toplantıda değil, ürünün kendisinde duruyor.

Muhammet Şafak · 3 Ekim 2026



02 İKİ MODELİN MEKANİĞİ

Gitflow ve GitHub Flow

KONU	GİTFLOW	GİTHUB FLOW
Branch'ler	main, develop, feature/*, release/*, hotfix/*	Tek main ve ondan dallanan kısa ömürlü branch'ler
Sürüm	Sürüm odaklı; her yayın için ayrı release branch	Sürüm kavramı yok; tek yaşayan sürüm prod'dakidir
Hotfix	Ayrı bir hotfix/* hattı	Sıraya giren bir PR
Doğal habitat	Mobil uygulama, SDK ve kütüphane, on-premise ya da packaged yazılım	Web uygulaması ve SaaS, iç araçlar, API servisleri

03 ÜÇ SORU, BİR VARSAYILAN

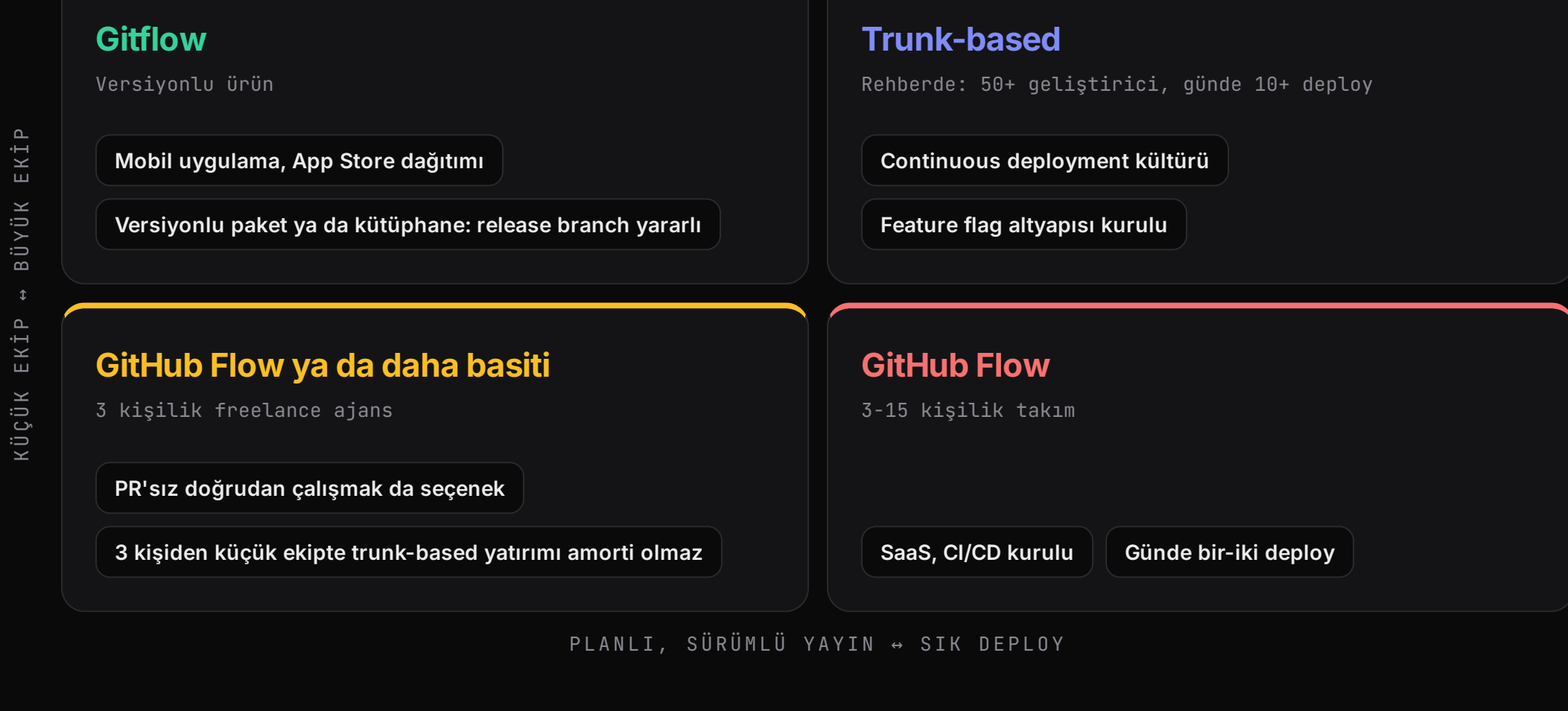
Branch stratejisini ürün seçer



04 EKİP VE DEPLOY TEMPOSU

Üç akış, dört durum

Dikey eksen ekip büyüklüğünü, yatay eksen deploy temposunu gösterir. Hücreler Production'da Git rehberinden ve trunk-based yazısından derlendi.



05 PRODUCTION'DA GİT REHBERİNDEN

Karar tablosu

DURUM	TAVSİYE
Mobil uygulama, App Store dağıtımı	Gitflow
SaaS, küçük-orta takım, günlük deploy	GitHub Flow
3 kişilik freelance ajans	GitHub Flow
50+ geliştirici, deploy günde 10+	Trunk-based
Versiyonlu paket ya da kütüphane	Gitflow
Plansız, “ne lazımsa o”	GitHub Flow

06 TRUNK-BASED'E NE ZAMAN

Yazılarda iki ayrı eşik

- TRUNK YAZISI

Deploy frekansı haftada 5+ olduğunda release branch bir düzen değil, bir engel olmaya başlıyor. Trunk-based'in savunulduğu projeler günde 3-10 deploy yapıyor.
- GİTFLOW YAZISI

Günde 5+ deploy'da GitHub Flow'un kısa ömürlü branch'leri bile sürtünme kaynağı oluyor; o eşikten sonra trunk-based.
- İPUCU

Haftada 1 deploy yapan bir ekip için geçmeye zorlayan bir sebep yok; Gitflow ya da GitHub Flow o ritimde sorunsuz çalışıyor.

07 TRUNK-BASED HEDEFLERİ

Tempoyu taşıyan rakamlar

5-10 dakika Hedef: merge'den prod'a pipeline süresi	1-2 gün Eşik: feature branch'in en uzun ömrü	200-400 satır Hedef: tek mantıksal değişiklikli PR boyu	60-90 dakika Bu boyutta bir PR'ın review süresi
--	---	--	--

08 GEÇMEDEN ÖNCE

Trunk-based'in dört temeli

- ☐ Yeşil trunk, sözleşme seviyesinde: CI ve zorunlu PR check'leri garanti eder
- ☐ Feature flag altyapısı: yarım iş koda girer, kullanıcıya kapalı kalır
- ☐ Hızlı pipeline: merge'den prod'a 5-10 dakika
- ☐ Kısa ömürlü branch kültürü: açıldığı gün ya da ertesi gün kapanır



09 TRUNK-BASED VE ÖLÇÜM

Geçince ölç, şu sahnelerde geçme

Yap	Yapma
Lead time for changes için hedef: p50 < 1 saat	Regüle sektörde zorunlu pre-prod sign-off varken geçme
Change failure rate için hedef: < %15	Aynı anda 3.x ve 4.x desteklenecekse geçme
Mean time to restore için hedef: < 30 dakika	CI yeşilini garantileyemeyen ekipte geçme
Metrikler kötüleşirse önce CI ve flag disiplinine bak	3 kişiden küçük, haftada 1 deploy yapan ekipte geçme

10 EŞLEŞME HATALARI

Modeli yanlış yere koymak

- ANTI-PATTERN

Üç akışı karıştırmak: develop branch, feature flag ve release branch bir arada. Üç ayrı disiplinin yansını uygulamak, hiçbirinin avantajını almamak demek.
- SAAS'TA GİTFLOW

develop gereksiz bir gölge prod olur; PR'lar iki kez merge edilir. Kazanılan izolasyon, ödenen sürtünmeye değmez.
- PACKAGED ÜRÜNDE GİTHUB FLOW

Eski sürümdeki müşteri hotfix istediğinde ekip olmayan bir bakım branch'ini arar.
- KARMA MODEL

İki modelden birini doğru uygulamak, üçüncü bir model uydurmaktan her zaman ucuzdur.

11 KARAR

Git workflow'unu ürüne göre seçin; ürünü workflow'a göre değil.

Seçim “hangisi modern” değil, “ürün şekliniz hangisini doğal kılıyor” sorusudur. Sürüm sayısı, müşteri tipi ya da deploy frekansı değiştiğinde karar yeniden açılır.