Build & Deployment

npm run build ile production build almak, Vite'ta ortam değişkenleri (VITE_ öneki, .env), ve tamamen statik bir React uygulamasını Vercel'e deploy etmek -- basit örneklerle.

Orta 13 dk

Son güncelleme: 18 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/build-deployment

---

Build & Deployment

Şimdiye kadar her uygulamayı `npm run dev` ile, yalnızca kendi bilgisayarımızda çalıştırdık. Bu ders, bir React uygulamasını gerçek bir production build'e dönüştürüp internete açık, herkesin erişebileceği bir adrese (**Vercel**) deploy etmeyi anlatıyor.

npm run build: Production Build Almak

`npm run dev` sırasında Vite, kodu anlık olarak (her değişiklikte yeniden) tarayıcıya gönderir -- bu, geliştirme için hızlıdır ama production için uygun değildir. `npm run build` çalıştırıldığında Vite:

```
npm run build
```

kodu küçültür (minify), gereksiz her şeyi ayıklar, ve bir `dist/` klasörüne statik `.html` / `.js` / `.css` dosyaları olarak yazar. Bu `dist/` klasörü, deploy edilecek TAM olarak budur -- bir sunucuya, React'in kendisine, hatta Node.js'e bile ihtiyaç duymadan, herhangi bir statik dosya sunucusu tarafından servis edilebilir.

Vite'ta Ortam Değişkenleri

Aynı kodun, yerelde ve production'da FARKLI değerlerle çalışması gerekebilir (bir API adresi, bir versiyon numarası gibi). Vite, `VITE_` öneki taşıyan ortam değişkenlerini build sırasında koda gömer:

```
// Vite only embeds environment variables that start with "VITE_" into the
// client code -- they're read via import.meta.env at build time. The value
// comes from the .env file (local development) or from Vercel's "Environment
// Variables" setting (production); if it's undefined, `??` falls back to a default.
const appVersion = import.meta.env.VITE_APP_VERSION ?? "dev";

function VersionBadge() {
  return <span className="badge">v{appVersion}</span>;
}

export default VersionBadge;
```

`.env` dosyasına `VITE_APP_VERSION=1.0.0` yazılırsa, `import.meta.env.VITE_APP_VERSION` bu değeri okur -- dosya yoksa (ya da değişken tanımlı değilse) `??` ile verilen `"dev"` varsayılanına düşülür. `VITE_` ÖNEKİ OLMAYAN değişkenler istemci koduna hiç gömülmez -- bu, yanlışlıkla gizli bir anahtarın tarayıcı koduna sızmasını önleyen bilinçli bir güvenlik kararı.

Ortama Göre Davranmak: Feature Flag'ler

Ortam değişkenleri yalnızca metin göstermek için değil, ortama göre farklı DAVRANMAK için de kullanılır:

```
// Environment variables aren't just for showing text/badges -- they're also
// used to behave differently depending on the environment (feature flags).
// It's important not to forget to convert the "true" string into an actual
// boolean -- every value in import.meta.env is a STRING, even "false" is truthy.
const showBetaBanner = import.meta.env.VITE_SHOW_BETA_BANNER === "true";

function BetaBanner() {
  if (!showBetaBanner) {
    return null;
  }

  return <div className="beta-banner">This is a beta build.</div>;
}

export default BetaBanner;
```

`import.meta.env` içindeki her değer bir STRING'dir -- `"false"` bile truthy'dir, bu yüzden `=== "true"` ile açıkça karşılaştırmak gerekir.

.env Dosyaları ve Git

`.env` dosyaları genellikle git'e COMMIT EDİLMEZ (`.gitignore`'a eklenir) -- gerçek değerler kişiye/ortama özel olabilir. Bunun yerine bir `.env.example` dosyası (gerçek değerler olmadan, hangi değişkenlerin gerektiğini gösteren) repoya eklenir; her geliştirici bunu kendi `.env`'ine kopyalar.

Vercel'e Deploy Etmek

Bir React/Vite projesini Vercel'e deploy etmenin adımları:

1. vercel.com üzerinde GitHub hesabınla giriş yap.
2. "Add New..." → "Project" seç, reponu bağla.
3. Bu bir monorepo ise (birden fazla proje aynı repoda), "Root Directory" alanına ilgili proje klasörünü yaz.
4. Vercel, Vite projelerini OTOMATİK tanır -- Build Command (`npm run build`) ve Output Directory (`dist`) senin için doğru ayarlanır.
5. "Environment Variables" bölümüne, `.env` 'indeki değişkenleri ekle.
6. "Deploy"e tıkla -- birkaç saniye içinde canlı bir URL alırsın.

`main` branch'ine her push, otomatik yeni bir production deploy tetikler; her Pull Request de kendi "preview" deploy'unu alır -- birleştirmeden önce değişikliği canlı bir URL'de görebilirsin.

Özet ve Terimler Sözlüğü

`npm run build` , bir React uygulamasını statik dosyalara (`dist/`) dönüştürür -- bu dosyalar herhangi bir statik sunucu tarafından servis edilebilir. Vite, `VITE_` önekli ortam değişkenlerini build sırasında koda gömer; `.env` dosyası yerel değerleri taşır ve genellikle git'e commit edilmez. Vercel, bir GitHub reposunu bağlayıp Vite projelerini otomatik tanıyarak, her push'ta yeni bir deploy tetikleyen bir platformdur.

Terimler Sözlüğü

Production Build — Bir uygulamanın, gerçek kullanıcılara sunulmak üzere küçültülmüş /optimize edilmiş hâli.

Ortam Değişkeni (Environment Variable) — Kodun dışında tutulan, ortama (yerel /production) göre değişebilen bir yapılandırma değeri.

Preview Deployment — Bir Pull Request için otomatik oluşturulan, geçici ve bağımsız bir deploy.

Pratik Proje

Bu derste öğrendiğimiz kavramları (production build, ortam değişkenleri, Vercel'e deploy) kullanan, gerçek ve ÇALIŞTIRILMIŞ bir örnek proje var: **Build & Deployment Demo** -- canlı hâli **burada**. Sayfanın başlığındaki rozet (`v1.0.0`), Vercel'in "Environment Variables" ayarına eklenen `VITE_APP_VERSION` değişkeninden geliyor.

```
git clone https://github.com/cdurgun/react-course-projects.git
cd react-course-projects
npm install
cp projects/build-deployment/.env.example projects/build-deployment/.env
cd projects/build-deployment
npm run dev
```

`react-course-projects` deposu **npm workspaces** kullanır -- `npm install` yalnızca bir kez, depo kökünde çalıştırılır ve tüm proje klasörleri ortak bağımlılıkları paylaşır (her klasörde ayrı bir `node_modules` oluşmaz).