Build & Deployment

Making a production build with npm run build, environment variables in Vite (VITE_ prefix, .env), and deploying a fully static React app to Vercel -- with simple examples.

Intermediate 13 min

Last updated: August 18, 2026
Read online: https://www.learnforgex.com/en/topics/build-deployment

---

Build & Deployment

So far we've run every app with `npm run dev`, only on our own machine. This lesson covers turning a React app into a real production build and deploying it to a public, internet-facing address (**Vercel**).

npm run build: Making a Production Build

During `npm run dev`, Vite sends code to the browser on the fly (re-sending on every change) -- great for development, but not suited for production. Running `npm run build`:

```
npm run build
```

minifies the code, strips out anything unnecessary, and writes static `.html` / `.js` / `.css` files into a `dist/` folder. This `dist/` folder is EXACTLY what gets deployed -- it can be served by any static file server, with no need for a backend server, React itself, or even Node.js.

Environment Variables in Vite

The same code may need to behave DIFFERENTLY locally vs. in production (an API address, a version number). Vite bakes environment variables prefixed with `VITE_` into the code at build time:

```
// 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;
```

If `.env` contains `VITE_APP_VERSION=1.0.0`, `import.meta.env.VITE_APP_VERSION` reads that value -- if the file (or variable) doesn't exist, it falls back to the `"dev"` default given via `??`. Variables WITHOUT the `VITE_` prefix are never bundled into the client code at all -- a deliberate security choice that prevents a secret key from accidentally leaking into browser code.

Behaving Differently by Environment: Feature Flags

Environment variables aren't just for displaying text -- they're also used to make the app BEHAVE differently depending on the environment:

```
// 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;
```

Every value inside `import.meta.env` is a STRING -- even `"false"` is truthy, so it needs to be compared explicitly with `=== "true"`.

.env Files and Git

`.env` files are usually NOT committed to git (they're added to `.gitignore`) -- the real values may be personal or environment-specific. Instead, a `.env.example` file (showing which variables are needed, without real values) is committed; each developer copies it into their own `.env`.

Deploying to Vercel

Steps to deploy a React/Vite project to Vercel:

1. Sign in at vercel.com with your GitHub account.
2. Choose "Add New..." → "Project", and connect your repo.
3. If this is a monorepo (multiple projects in one repo), set "Root Directory" to the specific project folder.
4. Vercel AUTOMATICALLY detects Vite projects -- the Build Command (`npm run build`) and Output Directory (`dist`) are set correctly for you.
5. Add the variables from your `.env` to the "Environment Variables" section.
6. Click "Deploy" -- you get a live URL within seconds.

Every push to `main` automatically triggers a new production deployment; every Pull Request also gets its own "preview" deployment -- you can see the change live before merging.

Summary and Glossary

`npm run build` turns a React app into static files (`dist/`) that can be served by any static server. Vite bakes `VITE_`-prefixed environment variables into the code at build time; the `.env` file holds local values and is usually not committed to git. Vercel is a platform that connects to a GitHub repo, automatically recognizes Vite projects, and triggers a new deployment on every push.

Glossary

Production Build — A minified/optimized version of an app, meant to be served to real users.

Environment Variable — A configuration value kept outside the code that can vary between environments (local/production).

Preview Deployment — A temporary, standalone deployment automatically created for a Pull Request.

Practical Project

There's a real project, using the concepts from this lesson (production build, environment variables, deploying to Vercel), that's actually been DEPLOYED: **Build & Deployment Demo** -- live at **this address**. The badge in the page header (`v1.0.0`) comes from the `VITE_APP_VERSION` variable added to Vercel's "Environment Variables" settings.

```
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
```

The `react-course-projects` repo uses **npm workspaces** -- `npm install` only needs to run once, at the repo root, and every project folder shares the same dependencies (no separate `node_modules` per folder).