Chapter 12: Deployment và các bước tiếp theo trong Next.js

A. Mục tiêu

Sau bài học này, học viên sẽ:

- Hiểu quy trình **build** và triển khai một ứng dụng Next.js cho môi trường **production**.
- Biết cách sử dụng và quản lý **biến môi trường (Environment Variables)** để lưu trữ thông tin cấu hình và bảo mật.
- Thành thạo việc **triển khai (deploy)** ứng dụng Next.js lên nền tảng **Vercel** với quy trình **Continuous Deployment**.
- Nắm được các hướng đi tiếp theo để phát triển kỹ năng trở thành một lập trình viên **React/Next.js** chuyên nghiệp.
- Hiểu các khái niệm nâng cao như **testing**, **internationalization (i18n)**, **App Router**, **authentication**, và **UI component libraries**.

B. Nội dung lý thuyết

1. Build cho Production

Để chuẩn bị ứng dụng Next.js cho môi trường production, bạn cần chạy lệnh build để tạo ra phiên bản tối ưu hóa của ứng dụng.

Quy trình build:

- Chạy lệnh:

```
npm run build
```

Hoặc:

```
next build
```

- Next.js thực hiện các tối ưu hóa:
  - **Static Generation (SSG)**: Chạy `getStaticProps` và `getStaticPaths` để tạo các trang tĩnh.
  - **Server-Side Rendering (SSR)**: Chuẩn bị các trang sử dụng `getServerSideProps`.
  - **Minification**: Nén code JavaScript và CSS để giảm kích thước.
  - **Image Optimization**: Tối ưu hóa hình ảnh sử dụng `next/image`.
  - **Font Optimization**: Tối ưu hóa font với `next/font`.
- Kết quả được lưu trong thư mục `.next`. Đây là thư mục chứa toàn bộ tài nguyên cần thiết để chạy ứng dụng ở production.

Chạy ứng dụng production cục bộ:

- Sau khi build, chạy:

```
npm run start
```

Hoặc:

```
next start
```

- Truy cập `http://localhost:3000` để kiểm tra ứng dụng ở chế độ production.

Lưu ý:

- Chế độ production không có **hot-reload** như chế độ development.
- Đảm bảo ứng dụng không có lỗi trước khi build (kiểm tra bằng `npm run dev`).

---

2. Biến môi trường (Environment Variables)

Biến môi trường được sử dụng để lưu trữ các thông tin cấu hình như **API keys**, **database credentials**, hoặc các giá trị khác nhau giữa các môi trường (development, production).

Cách sử dụng trong Next.js:

1. Tạo file `.env.local` trong thư mục gốc của dự án:

```
DB_HOST=localhost
DB_USER=myuser
NEXT_PUBLIC_API_URL=https://api.example.com
```

2. **Phân loại biến môi trường**:
   - **Biến server-side**: Các biến như `DB_HOST`, `DB_USER` chỉ có thể truy cập trong code chạy trên server (như `getStaticProps`, `getServerSideProps`, hoặc API Routes).
   - **Biến client-side**: Các biến bắt đầu bằng `NEXT_PUBLIC_` (ví dụ: `NEXT_PUBLIC_API_URL`) có thể truy cập trong code chạy trên client (như trong component).
3. Truy cập biến môi trường trong code:

```
// Server-side
export async function getStaticProps() {
  const dbHost = process.env.DB_HOST;
  return { props: {} };
}

// Client-side
```

```
export default function Home() {
  const apiUrl = process.env.NEXT_PUBLIC_API_URL;
  return <div>API URL: {apiUrl}</div>;
}
```

Lưu ý:

- File `.env.local` không nên commit lên Git (thêm vào `.gitignore`).
- Các biến `NEXT_PUBLIC_` sẽ được nhúng vào bundle JavaScript, nên tránh lưu thông tin nhạy cảm trong đó.
- Khi deploy lên Vercel, cần cấu hình lại biến môi trường trên giao diện Vercel.

---

3. Deployment với Vercel

Vercel là nền tảng được tạo bởi đội ngũ phát triển Next.js, cung cấp trải nghiệm triển khai đơn giản, nhanh chóng và tối ưu cho các ứng dụng Next.js.

Quy trình triển khai:

1. **Đẩy code lên Git repository**:
    - Tạo repository trên GitHub, GitLab, hoặc Bitbucket.
    - Đẩy code dự án lên:

```
git init
git add .
git commit -m "Initial commit"
git remote add origin <repository-url>
git push -u origin main
```

2. **Tạo tài khoản Vercel**:
    - Đăng ký tại vercel.com (có thể dùng tài khoản GitHub để đăng nhập).

3. **Tạo dự án mới trên Vercel**:
    - Trên dashboard Vercel, nhấn “New Project”.
    - Chọn repository Git của bạn.
    - Vercel tự động phát hiện dự án là Next.js và cấu hình build settings.
    - Nhấn “Deploy” để bắt đầu triển khai.
4. **Continuous Deployment**:
    - Mỗi khi bạn đẩy code mới lên nhánh chính (main/master), Vercel sẽ tự động build và deploy lại.
    - Vercel cung cấp URL cho ứng dụng (ví dụ: `https://my-app.vercel.app`).
5. **Cấu hình biến môi trường trên Vercel**:
    - Vào phần Settings → Environment Variables của dự án.
    - Thêm các biến như `NEXT_PUBLIC_API_URL`, `DB_HOST`.
    - Redeploy để áp dụng thay đổi.

Ưu điểm của Vercel:

- Tích hợp chặt chẽ với Next.js, hỗ trợ SSG, SSR, và API Routes.
- Tự động tối ưu hóa hiệu năng (CDN, image optimization).
- Hỗ trợ preview deployments cho các pull requests.
- Miễn phí cho các dự án cá nhân và nhỏ.

Lưu ý:

- Đảm bảo file `.env.local` không được đẩy lên Git.
- Kiểm tra log build trên Vercel để phát hiện lỗi triển khai.

---

4. Các bước tiếp theo để trở thành lập trình viên React/Next.js chuyên nghiệp

Sau khi nắm vững các kiến thức cơ bản, bạn cần học thêm các kỹ năng nâng cao để phát triển sự nghiệp:

a. Testing

- **Unit Testing và Integration Testing**:
  - Sử dụng **Jest** và **React Testing Library** để viết test cho component và logic.
  - Ví dụ:

```
import { render, screen } from '@testing-library/react';
import MyComponent from './MyComponent';

test('renders component', () => {
  render(<MyComponent />);
  expect(screen.getByText('Hello')).toBeInTheDocument();
});
```

- **End-to-End (E2E) Testing**:
  - Sử dụng **Cypress** hoặc **Playwright** để kiểm tra luồng người dùng.
  - Ví dụ (Cypress):

```
describe('Contact Form', () => {
  it('submits form successfully', () => {
    cy.visit('/contact');
    cy.get('input[name="name"]').type('John Doe');
    cy.get('button[type="submit"]').click();
    cy.contains('Form submitted').should('be.visible');
  });
});
```

b. Internationalization (i18n)

- Hỗ trợ đa ngôn ngữ với thư viện **next-i18next**.
- Cấu hình cơ bản:

```
// next.config.js
module.exports = {
  i18n: {
    locales: ['en', 'vi'],
    defaultLocale: 'en',
  },
};
```

- Sử dụng trong component:

```
import { useTranslation } from 'next-i18next';

export default function Home() {
  const { t } = useTranslation('common');
  return <h1>{t('welcome')}</h1>;
}
```

c. App Router

- Tìm hiểu **App Router** (thư mục `app`) trong Next.js, thay thế cho Pages Router.
- Hỗ trợ **Server Components**, **Layouts**, và **Streaming**.
- Ví dụ cấu trúc thư mục:

```
app/
  layout.js
  page.js
  [slug]/
    page.js
```

- Ví dụ `layout.js`:

```
export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body>
        <nav>Navigation</nav>
        {children}
      </body>
    </html>
  );
}
```

d. Authentication

- Tích hợp xác thực với **NextAuth.js** hoặc **Clerk**.
- Ví dụ với NextAuth.js:

```
// pages/api/auth/[...nextauth].js
import NextAuth from 'next-auth';
import GitHubProvider from 'next-auth/providers/github';

export default NextAuth({
  providers: [
    GitHubProvider({
      clientId: process.env.GITHUB_ID,
      clientSecret: process.env.GITHUB_SECRET,
    }),
  ],
});
```

e. UI Component Libraries

- Làm việc với các thư viện như **Material-UI**, **Chakra UI**, hoặc **Shadcn/ui** để xây dựng giao diện nhanh chóng.
- Ví dụ với Material-UI:

```
import Button from '@mui/material/Button';

export default function Home() {
  return <Button variant="contained">Click Me</Button>;
}
```

C. Bài tập thực hành

Bài 1: Build và chạy ứng dụng ở chế độ production

1. Chạy lệnh build:

```
npm run build
```

2. Kiểm tra output trong thư mục `.next`.
3. Chạy ứng dụng production cục bộ:

```
npm run start
```

4. Truy cập `http://localhost:3000` và kiểm tra ứng dụng.

Bài 2: Sử dụng biến môi trường

1. Tạo file `.env.local`:

```
NEXT_PUBLIC_APP_TITLE="My Awesome Blog"
```

2. Sử dụng biến trong component `components/Layout.js`:

```
export default function Layout({ children }) {
  const title = process.env.NEXT_PUBLIC_APP_TITLE;
  return (
    <div>
      <header>
        <h1>{title}</h1>
      </header>
      <main>{children}</main>
    </div>
  );
}
```

3. Kiểm tra: Chạy `npm run dev` và xác nhận tiêu đề hiển thị đúng.

Bài 3: Đẩy code lên GitHub

1. Tạo repository mới trên GitHub.
2. Đẩy code dự án:

```
git init
git add .
git commit -m "Initial commit"
git remote add origin <repository-url>
git push -u origin main
```

3. Đảm bảo file `.env.local` được thêm vào `.gitignore`.

Bài 4: Deploy lên Vercel

1. Đăng ký tài khoản Vercel tại vercel.com.
2. Tạo dự án mới:

    - Kết nối với repository GitHub vừa tạo.
    - Nhấn “Deploy”.
3. Truy cập URL được cung cấp (ví dụ: `https://my-app.vercel.app`).
4. Chia sẻ URL với giảng viên hoặc đồng nghiệp.

Bài 5: Cấu hình biến môi trường trên Vercel

1. Truy cập phần Settings → Environment Variables của dự án trên Vercel.
2. Thêm biến:

```
Name: NEXT_PUBLIC_APP_TITLE
Value: My Production Blog
```

3. Redeploy dự án:
    - Đẩy một commit mới lên GitHub (ví dụ: thay đổi nhỏ trong README).
    - Hoặc nhấn “Redeploy” trên Vercel.
4. Kiểm tra URL ứng dụng và xác nhận tiêu đề đã thay đổi thành “My Production Blog”.

D. Bổ sung: Tối ưu hóa và giám sát sau triển khai

1. Tối ưu hóa hiệu năng

- Sử dụng **Vercel Analytics** để theo dõi hiệu năng ứng dụng:
    - Truy cập tab Analytics trên dashboard Vercel.
    - Kiểm tra các chỉ số như **First Contentful Paint (FCP)**, **Largest Contentful Paint (LCP)**, và **Cumulative Layout Shift (CLS)**.
- Tối ưu hóa thêm:
    - Sử dụng `next/image` với thuộc tính `priority` cho hình ảnh quan trọng.

- Lazy-load các component nặng bằng `next/dynamic`.
- Giảm kích thước font bằng cách chỉ tải các `subsets` cần thiết trong `next/font`.

2. Giám sát lỗi

- Tích hợp **Sentry** hoặc **Vercel Log Drains** để theo dõi lỗi runtime:

```
// pages/_app.js
import * as Sentry from '@sentry/nextjs';

Sentry.init({
  dsn: process.env.SENTRY_DSN,
});
```

- Kiểm tra log trên Vercel (tab Deployments → Logs) để phát hiện lỗi build hoặc runtime.

3. Tối ưu hóa SEO

- Thêm **meta tags** và **Open Graph** để cải thiện SEO và chia sẻ trên mạng xã hội:

```
// pages/_document.js
import Document, { Html, Head, Main, NextScript } from 'next/document';

export default class MyDocument extends Document {
  render() {
    return (
      <Html lang="vi">
        <Head>
          <meta name="description" content="My Awesome Blog" />
          <meta property="og:title" content={process.env.NEXT_PUBLIC_APP
        </Head>
        <body>
          <Main />
          <NextScript />
        </body>
      </Html>
```

```
    );
  }
}
```

E. So sánh các nền tảng triển khai

Nền tảng	Ưu điểm	Nhược điểm
Vercel	Tích hợp chặt chẽ với Next.js, dễ sử dụng, hỗ trợ SSG/SSR, CDN mạnh mẽ	Chi phí cao cho các dự án lớn, hạn chế tùy chỉnh server
Netlify	Hỗ trợ SSG tốt, dễ tích hợp với CMS, miễn phí cho dự án nhỏ	Hỗ trợ SSR hạn chế, không tối ưu cho Next.js như Vercel
Heroku	Linh hoạt, hỗ trợ Node.js server, dễ mở rộng	Build chậm, không tối ưu cho SSG, chi phí tăng khi sử dụng add-ons
AWS Amplify	Tích hợp tốt với AWS, phù hợp cho dự án lớn	Phức tạp để cấu hình, không thân thiện với người mới

F. Tài liệu tham khảo

- Next.js Documentation - Deployment
- Vercel Documentation
- Next.js Environment Variables
- Jest Documentation
- NextAuth.js Documentation

G. Bài tập nâng cao (tùy chọn)

1. **Tích hợp Sentry**:
   - Cài đặt `@sentry/nextjs` và cấu hình để theo dõi lỗi.
   - Tạo một lỗi giả (throw new Error) và kiểm tra lỗi trên dashboard Sentry.
2. **Tùy chỉnh domain**:
   - Mua một domain và cấu hình trên Vercel để sử dụng domain tùy chỉnh (ví dụ: `myblog.com`).
3. **Tích hợp i18n**:
   - Cấu hình `next-i18next` để hỗ trợ tiếng Anh và tiếng Việt.
   - Tạo file dịch: `public/locales/en/common.json` và `public/locales/vi/common.json`.
4. **Viết test**:
   - Viết unit test cho component `Layout` bằng Jest và React Testing Library.
   - Viết E2E test cho trang `/contact` bằng Cypress.
5. **Tối ưu hóa SEO**:
   - Thêm `next-seo` để quản lý meta tags động.
   - Cấu hình sitemap và robots.txt trên Vercel.