Docker Setup Guide for Next.js Application

Environment: Next.js on Windows 10/11

This comprehensive guide will walk you through setting up Docker for a Next.js application on Windows 10, from installation to deployment.

Prerequisites

Before starting, ensure you have:

- ☑ Administrator access on your machine
- ☑ Node.js installed locally (for development)
- ☑ A Next.js project (existing or new)

Step 1: Install Docker Desktop

Download and Installation

1. Download Docker Desktop

- Visit Docker Desktop for Windows
- Click **"Download for Windows"**
- Choose the appropriate version for your system (AMD64 if you have Intel or AMD processor)

2. Run the Installer

- Execute the downloaded `.exe` file as **Administrator**
- **Important:** Enable **WSL 2 backend** when prompted (recommended for better performance)
- Follow the installation wizard prompts

3. Post-Installation Setup

- Restart your computer when prompted
- Launch **Docker Desktop** from the Start menu
- Wait for Docker to start completely (the whale icon in the system tray should be steady)
- When first opening Docker Desktop, it will prompt to update WSL to recent version

4. Update WSL 2 (if prompted)

If Docker does not ask to update WSL, skip this step.

To update WSL 2:

- Open **Command Prompt** by typing `cmd` in the search bar
- Type the following command:

```
wsl --update
```

5. Verify Installation

Open Command Prompt and run:

```
docker --version
docker-compose --version
```

You should see version numbers for both Docker and Docker Compose.

Step 2: Prepare Your Next.js Project Structure

Your Next.js project should have the following structure:

```
my-nextjs-app/
├── .next/                    # Next.js build output (generated)
├── node_modules/             # Dependencies (generated)
├── public/                   # Static assets
├── src/                      # Source code (or pages/ in root)
│   ├── app/                  # App Router (Next.js 13+)
│   │   ├── layout.tsx
│   │   └── page.tsx
│   └── components/           # React components
├── .dockerignore            # Files to exclude from Docker
├── .env                      # Environment variables (keep private)
├── .env.example              # Environment template (optional)
├── .env.local                # Local environment variables
├── .gitignore               # Git ignore file
├── docker-compose.yml       # Docker Compose configuration
├── Dockerfile               # Docker build instructions
├── next.config.js           # Next.js configuration
├── package.json             # Node.js dependencies
├── package-lock.json        # Locked dependencies
└── tsconfig.json            # TypeScript configuration (if using TS)
```

Step 3: Create Docker Configuration Files

Create these 2 new files in your project root directory:

1. `.dockerignore`

This file tells Docker which files and folders to exclude from the build context:

```
node_modules
npm-debug.log
Dockerfile
.dockerignore
.next
.git
.gitignore
```

2. `Dockerfile`

This file defines how your Next.js application container is built:

```dockerfile
# Step 1: Build stage
FROM node:18-alpine AS builder

# Set working directory
WORKDIR /app

# Copy dependency files and install
COPY package*.json ./
RUN npm install

# Copy all project files
COPY . .

# Build Next.js app
RUN npm run build

# Step 2: Run stage
FROM node:18-alpine AS runner

WORKDIR /app

# Copy only needed build output
COPY --from=builder /app/.next ./.next
COPY --from=builder /app/public ./public
COPY --from=builder /app/package*.json ./

# Install only production dependencies
RUN npm install --omit=dev

# Expose port
EXPOSE 3000

# Start the app
CMD ["npm", "start"]
```

Step 3: Update package.json Scripts

Add Docker-friendly scripts to your `package.json`:

```
{
  "name": "my-nextjs-app",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "docker:build": "docker-compose build",
    "docker:up": "docker-compose up -d",
    "docker:down": "docker-compose down",
    "docker:logs": "docker-compose logs -f",
    "docker:restart": "docker-compose restart"
  }
}
```

Step 4: Build and Run Your Docker Container

Option 1: Using Docker Compose (Recommended)

Build

```
> docker build -t my-nextjs-app .
```

Start the container:

```
docker run -p 3000:3000 my-nextjs-app
```