# 🐳 Spotify MVP - Docker Deployment Guide

Deploy your Spotify MVP using Docker with local source file mounting for development.

## 🎯 Two Docker Deployment Methods

### **Method 1: Development with Local Source Files** (Recommended for Development)

Your local files are mounted as volumes - changes reflect immediately without rebuilding.

### **Method 2: Production with Built Containers**

Files are built into containers - for production deployment.

## 🚀 Method 1: Development with Local Source Files

### **Quick Start**

# 1. Navigate to project directory  
cd spotify-mvp  
  
# 2. Use the development Docker Compose  
docker-compose -f ../docker-compose.dev.yml up --build  
  
# Your app will be available at:  
# Frontend: http://localhost:3000  
# Backend API: http://localhost:3001/api

### **What This Method Gives You:**

✅ **Live Reloading** - Changes to local files immediately reflect in containers  
✅ **Fast Development** - No need to rebuild containers for code changes  
✅ **Node Modules Persistence** - Dependencies are cached in named volumes  
✅ **Database Persistence** - PostgreSQL data persists between restarts

### **Development Workflow:**

1. Edit files in your local backend/ or frontend/ directories
2. Changes are automatically detected and the servers restart
3. View changes immediately in your browser

## 🏗️ Method 2: Production with Built Containers

### **Quick Start**

# Navigate to project directory  
cd spotify-mvp  
  
# Build and start containers  
docker-compose up --build  
  
# Your app will be available at:  
# Frontend: http://localhost:3000  
# Backend API: http://localhost:3001/api

### **What This Method Gives You:**

✅ **Production Ready** - Optimized containers for deployment  
✅ **Self Contained** - All code built into containers  
✅ **Portable** - Can be deployed anywhere Docker runs

## 📁 Project Structure for Docker

Your project directory should look like this:

spotify-mvp/  
├── backend/ # Backend source code  
│ ├── src/ # API routes and logic  
│ ├── database/ # DB migrations  
│ ├── package.json # Backend dependencies  
│ ├── Dockerfile # Backend container config  
│ └── uploads/ # Audio file storage  
├── frontend/ # Frontend source code  
│ ├── src/ # React components  
│ ├── public/ # Static assets  
│ ├── package.json # Frontend dependencies  
│ └── Dockerfile # Frontend container config  
├── database/ # Database schema  
│ └── schema.sql # Initial DB schema  
├── sample-music/ # Demo audio files  
├── docker-compose.yml # Production Docker config  
└── docker-compose.dev.yml # Development Docker config (in parent dir)

## ⚙️ Docker Configuration Files

### **docker-compose.dev.yml** - Development Setup

# Uses volume mounts for live development  
volumes:  
 - ./backend:/app # Mount local backend code  
 - ./frontend:/app # Mount local frontend code  
 - backend\_node\_modules:/app/node\_modules # Cache dependencies

### **docker-compose.yml** - Production Setup

# Builds code into containers  
build:   
 context: ./backend  
 dockerfile: Dockerfile

## 🔧 Environment Configuration

### **Backend Environment Variables**

NODE\_ENV=development  
PORT=3001  
DATABASE\_URL=postgresql://spotify\_user:spotify\_password@postgres:5432/spotify\_mvp  
JWT\_SECRET=your-super-secure-jwt-secret  
CORS\_ORIGIN=http://localhost:3000  
UPLOAD\_PATH=./uploads

### **Frontend Environment Variables**

REACT\_APP\_API\_URL=http://localhost:3001/api  
REACT\_APP\_STREAM\_URL=http://localhost:3001/api/stream  
REACT\_APP\_APP\_NAME=Spotify MVP

## 🎵 Adding Your Music Files

### **Option 1: Use Sample Music**

Sample music is automatically available in containers:

# Files from ./sample-music/ are mounted to containers

### **Option 2: Add Your Own Music**

# Add your .mp3 files to the uploads directory  
mkdir -p backend/uploads  
cp your-music-files/\* backend/uploads/  
  
# They'll be available in the running containers immediately

## 🗄️ Database Setup

### **Automatic Database Initialization**

The PostgreSQL container automatically: 1. Creates the spotify\_mvp database 2. Runs database/schema.sql to create tables 3. Sets up proper users and permissions

### **Manual Database Operations**

# Connect to database  
docker exec -it spotify-mvp-db psql -U spotify\_user -d spotify\_mvp  
  
# Run SQL commands  
INSERT INTO artists (name, genre) VALUES ('Your Artist', 'Rock');

## 🔍 Monitoring & Debugging

### **View Logs**

# View all logs  
docker-compose logs -f  
  
# View specific service logs  
docker-compose logs -f backend  
docker-compose logs -f frontend  
docker-compose logs -f postgres

### **Health Checks**

# Check container status  
docker-compose ps  
  
# Test API health  
curl http://localhost:3001/health  
  
# Test database connection  
docker exec spotify-mvp-db pg\_isready -U spotify\_user

### **Access Container Shells**

# Backend container  
docker exec -it spotify-mvp-backend sh  
  
# Frontend container   
docker exec -it spotify-mvp-frontend sh  
  
# Database container  
docker exec -it spotify-mvp-db psql -U spotify\_user -d spotify\_mvp

## 🚀 Docker Commands Cheat Sheet

### **Starting Services**

# Development (with local files)  
docker-compose -f ../docker-compose.dev.yml up  
  
# Production (built containers)  
docker-compose up  
  
# Build and start (rebuild containers)  
docker-compose up --build  
  
# Run in background  
docker-compose up -d

### **Stopping Services**

# Stop all services  
docker-compose down  
  
# Stop and remove volumes  
docker-compose down -v  
  
# Stop and remove everything  
docker-compose down -v --rmi all

### **Managing Containers**

# View running containers  
docker ps  
  
# View all containers  
docker ps -a  
  
# Remove stopped containers  
docker container prune  
  
# Remove unused images  
docker image prune

## 🔧 Troubleshooting

### **Port Already in Use**

# Check what's using port 3000/3001  
lsof -i :3000  
lsof -i :3001  
  
# Stop conflicting processes  
kill -9 [PID]

### **Container Build Issues**

# Clear Docker cache  
docker system prune -a  
  
# Rebuild from scratch  
docker-compose build --no-cache

### **Database Connection Issues**

# Check PostgreSQL container  
docker logs spotify-mvp-db  
  
# Restart database only  
docker-compose restart postgres

### **Permission Issues (Linux/Mac)**

# Fix file permissions  
chmod -R 755 backend/  
chmod -R 755 frontend/  
  
# Fix ownership  
chown -R $USER:$USER .

## 🎯 Development Tips

### **Hot Reloading**

* Backend: Uses nodemon - restarts on file changes
* Frontend: Uses Vite dev server - hot module replacement
* Both automatically detect changes to mounted local files

### **Debugging**

# Add debug logs to containers  
docker-compose -f ../docker-compose.dev.yml up  
  
# View real-time logs  
docker-compose logs -f backend

### **Adding Dependencies**

When you add new npm packages locally:

# Remove node\_modules volume to refresh  
docker-compose down  
docker volume rm spotify-mvp\_backend\_node\_modules  
docker volume rm spotify-mvp\_frontend\_node\_modules  
  
# Restart to reinstall  
docker-compose -f ../docker-compose.dev.yml up --build

## 🎉 Success Indicators

When everything is working:

✅ **Database**: spotify-mvp-db container healthy  
✅ **Backend**: Available at http://localhost:3001/health  
✅ **Frontend**: Available at http://localhost:3000  
✅ **API**: Responds to http://localhost:3001/api/tracks  
✅ **Logs**: No error messages in container logs

## 🎵 Ready to Stream!

Your Spotify MVP is now running in Docker with your local source files mounted. You can:

* Make changes to code and see them immediately
* Add your own music files to backend/uploads/
* Create playlists and test the streaming functionality
* Deploy to any Docker-compatible hosting platform

**Access your app at:** http://localhost:3000

**Happy coding and streaming!** 🎶