🐳 Docker Deployment Guide

Enhanced Markdown Editor

This guide explains how to containerize and deploy your enhanced markdown editor using Docker.

📦 What's Included

- **Dockerfile**: Multi-stage build with nginx alpine for lightweight deployment
- **nginx.conf**: Optimized nginx configuration with security headers and compression
- **docker-compose.yml**: Complete orchestration setup with health checks
- **Build/Run scripts**: Convenient automation scripts
- **.dockerignore**: Optimized build context

🚀 Quick Start

Option 1: Using Docker Compose (Recommended)

```
# Build and start the container
docker-compose up -d

# Access the editor
open http://localhost:8080
```

Option 2: Using Docker Commands

```
# Build the image
docker build -t enhanced-markdown-editor .

# Run the container
docker run -d -p 8080:80 --name markdown-editor enhanced-markdown-
editor

# Access the editor
open http://localhost:8080
```

Option 3: Using Convenience Scripts

```
# Make scripts executable (if needed)
chmod +x build.sh run.sh

# Build the Docker image
./build.sh

# Run the container
./run.sh
```

🔧 Detailed Instructions

Building the Docker Image

1. **Navigate to the project directory**:
   `bash cd /workspace`
2. **Build the image**:
   `bash docker build -t enhanced-markdown-editor:latest .`

3. **Verify the build**:
   `bash docker images | grep enhanced-markdown-editor`

Running the Container

1. **Start the container**:
   `bash docker run -d \ --name enhanced-markdown-editor \ -p 8080:80 \ --restart unless-stopped \ enhanced-markdown-editor:latest`
2. **Access the application**:
   - Open your browser and go to `http://localhost:8080`
   - The enhanced markdown editor will be available immediately
3. **Check container status**:
   `bash docker ps`

Using Docker Compose

1. **Start the service**:
   `bash docker-compose up -d`
2. **Check service status**:
   `bash docker-compose ps`
3. **View logs**:
   `bash docker-compose logs -f`
4. **Stop the service**:
   `bash docker-compose down`

🔍 Container Details

Image Specifications

- **Base Image**: `nginx:alpine` (lightweight, ~40MB)
- **Architecture**: Multi-platform support
- **Security**: Includes security headers and CSP policies

- **Performance**: Gzip compression enabled

Port Configuration

- **Container Port**: 80 (nginx default)
- **Host Port**: 8080 (configurable)
- **Health Check**: Available at `/health`

Environment Variables

- `NGINX_HOST`: Server hostname (default: localhost)
- `NGINX_PORT`: Internal nginx port (default: 80)

🛠️ Customization Options

Custom Port

```
# Run on port 3000 instead of 8080
docker run -d -p 3000:80 enhanced-markdown-editor
```

Custom Configuration

```
# Mount custom nginx config
docker run -d \
  -p 8080:80 \
  -v $(pwd)/custom-nginx.conf:/etc/nginx/nginx.conf \
  enhanced-markdown-editor
```

Persistent Data (if needed)

```
# Mount a volume for any future file storage needs
docker run -d \
  -p 8080:80 \
  -v markdown-data:/app/data \
  enhanced-markdown-editor
```

🔄 Management Commands

Container Operations

```
# Start container
docker start enhanced-markdown-editor

# Stop container
docker stop enhanced-markdown-editor

# Restart container
docker restart enhanced-markdown-editor

# Remove container
docker rm enhanced-markdown-editor

# View logs
docker logs enhanced-markdown-editor

# Execute shell in container
docker exec -it enhanced-markdown-editor sh
```

Image Operations

```
# List images
docker images | grep enhanced-markdown-editor

# Remove image
docker rmi enhanced-markdown-editor

# Pull from registry (if published)
docker pull enhanced-markdown-editor
```

🌐 Production Deployment

Using Reverse Proxy (Nginx/Apache)

```
server {
    listen 80;
    server_name yourdomain.com;

    location / {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Using Traefik (Docker Labels Included)

The docker-compose.yml includes Traefik labels for automatic discovery:

```
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.markdown-editor.rule=Host(`markdown-editor.localhost`)"
```

Health Monitoring

```
# Check health status
curl http://localhost:8080/health

# Monitor container health
docker inspect enhanced-markdown-editor | grep Health -A 10
```

🐛 Troubleshooting

Common Issues

1. **Port already in use**:
   `bash # Use different port docker run -p 3000:80 enhanced-markdown-editor`
2. **Container won't start**:
   ```bash
   # Check logs
   docker logs enhanced-markdown-editor
   ```

Check if image exists
docker images | grep enhanced-markdown-editor
```

1. **Build fails**:
   ```bash
   # Clean Docker cache
   docker system prune -f

# Rebuild without cache
docker build --no-cache -t enhanced-markdown-editor .
```

Debug Mode

```
# Run container interactively for debugging
docker run -it --rm -p 8080:80 enhanced-markdown-editor sh
```

📊 Performance Optimization

Image Size Optimization

- Uses Alpine Linux base (minimal footprint)
- Multi-stage builds (if needed for future enhancements)
- Proper .dockerignore to exclude unnecessary files

Runtime Performance

- Nginx with optimized configuration
- Gzip compression enabled
- Static file caching headers
- Health check endpoint
```

🚀 Publishing to Registry

Docker Hub

```
# Tag for Docker Hub
docker tag enhanced-markdown-editor:latest yourusername/enhanced-markdown-editor:latest

# Push to Docker Hub
docker push yourusername/enhanced-markdown-editor:latest
```

Private Registry

```
# Tag for private registry
docker tag enhanced-markdown-editor:latest your-registry.com/enhanced-markdown-editor:latest

# Push to private registry
docker push your-registry.com/enhanced-markdown-editor:latest
```

📋 Summary

Your enhanced markdown editor is now fully containerized with:

✅ **Lightweight nginx-based container**
✅ **Production-ready configuration**
✅ **Health checks and monitoring**
✅ **Easy deployment options**
✅ **Security headers included**
✅ **Performance optimizations**

The application will be accessible at `http://localhost:8080` and ready for production deployment!