vitalDSP Webapp Deployment Guide

This guide provides comprehensive instructions for deploying the vitalDSP webapp to a server using Docker.

🚀 Quick Start

Prerequisites

- Docker (version 20.10 or higher)
- Docker Compose (version 2.0 or higher)
- At least 2GB RAM and 10GB disk space
- Linux server (Ubuntu 20.04+ recommended)

One-Command Deployment

```
# Clone the repository
git clone https://github.com/Oucru-Innovations/vital-DSP
cd vital-DSP

# Make deployment script executable
chmod +x deploy.sh

# Deploy the application
./deploy.sh
```

The application will be available at `http://vital-xxx.oucru.org:8000`

📋 Detailed Deployment Steps

1. Server Preparation

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Install Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Install Docker Compose
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Add user to docker group
```

```
sudo usermod -aG docker $USER
newgrp docker
```

2. Application Deployment

```
# Clone the repository
git clone https://github.com/Oucru-Innovations/vital-DSP
cd vital-DSP

# Deploy using the script
./deploy.sh deploy
```

3. Verify Deployment

```
# Check application status
./deploy.sh status

# Check logs
./deploy.sh logs

# Test health endpoint
curl http://localhost:8000/api/health
```

🐳 Docker Configuration

Production Dockerfile

The `Dockerfile.production` includes:

- **Multi-stage build** for optimized image size
- **Non-root user** for security
- **Health checks** for monitoring
- **Optimized caching** for faster builds

Docker Compose

The `docker-compose.yml` provides:

- **Application service** with proper configuration
- **Nginx reverse proxy** for production
- **Volume mounts** for persistent data
- **Health checks** and restart policies

🔧 Configuration Options

Environment Variables

Variable	Default	Description
`PORT`	8000	Application port
`PYTHONPATH`	/app:/app/src	Python path
`PYTHONUNBUFFERED`	1	Python output buffering

Port Configuration

```
# Change port
export PORT=8080
./deploy.sh deploy
```

Volume Mounts

The following directories are mounted for persistence:

- `./uploads` → `/app/uploads` (uploaded files)
- `./logs` → `/app/logs` (application logs)

🌐 Production Setup with Nginx

1. Enable Nginx Reverse Proxy

```
# Start with Nginx
docker-compose up -d nginx vitaldsp-webapp
```

2. Configure SSL (Optional)

```
# Create SSL directory
mkdir -p ssl

# Add your SSL certificates
cp your-cert.pem ssl/cert.pem
cp your-key.pem ssl/key.pem

# Uncomment HTTPS section in nginx.conf
# Restart containers
docker-compose restart nginx
```

3. Domain Configuration

Update `nginx.conf` with your domain:

```
server_name your-domain.com;
```

Monitoring and Maintenance

Health Checks

```
# Application health
curl http://localhost:8000/api/health

# Container health
docker-compose ps
```

Logs

```
# View all logs
./deploy.sh logs

# View specific service logs
docker-compose logs vitaldsp-webapp
docker-compose logs nginx
```

Updates

```
# Update application
./deploy.sh update

# Or manually
docker-compose down
docker-compose build --no-cache
docker-compose up -d
```

Security Considerations

1. Firewall Configuration

```
# Allow only necessary ports
sudo ufw allow 22    # SSH
sudo ufw allow 80    # HTTP
sudo ufw allow 443   # HTTPS
sudo ufw enable
```

2. SSL/TLS Setup

- Use Let's Encrypt for free SSL certificates
- Configure proper SSL ciphers in nginx.conf
- Enable HSTS headers

3. Container Security

- Application runs as non-root user
- Minimal base image (Python slim)
- No unnecessary packages installed

Troubleshooting

Common Issues

1. Port Already in Use

```
# Check what's using the port
sudo netstat -tulpn | grep :8000

# Kill the process or change port
export PORT=8080
./deploy.sh deploy
```

2. Permission Denied

```
# Fix directory permissions
sudo chown -R $USER:$USER uploads logs

# Recreate containers
docker-compose down
docker-compose up -d
```

3. Out of Memory

```
# Check memory usage
docker stats

# Increase swap space
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
```

```
sudo mkswap /swapfile
sudo swapon /swapfile
```

4. Application Not Starting

```
# Check logs
./deploy.sh logs

# Check container status
docker-compose ps

# Restart services
./deploy.sh restart
```

Debug Mode

```
# Run in debug mode
docker-compose -f docker-compose.yml -f docker-compose.debug.yml up
```

Performance Optimization

1. Resource Limits

Add to `docker-compose.yml`:

```
services:
  vitaldsp-webapp:
    deploy:
      resources:
        limits:
          memory: 2G
          cpus: '1.0'
```

2. Caching

- Nginx is configured with gzip compression
- Static files are cached for 1 year
- API responses are not cached

3. Database (Future)

For production with high load, consider adding:

- PostgreSQL database

- Redis for caching
- Load balancer for multiple instances

🔄 Backup and Recovery

Backup

```
# Backup uploads
tar -czf uploads-backup-$(date +%Y%m%d).tar.gz uploads/

# Backup logs
tar -czf logs-backup-$(date +%Y%m%d).tar.gz logs/
```

Recovery

```
# Restore uploads
tar -xzf uploads-backup-YYYYMMDD.tar.gz

# Restore logs
tar -xzf logs-backup-YYYYMMDD.tar.gz

# Restart application
./deploy.sh restart
```

📞 Support

For deployment issues:

1. Check the logs: `./deploy.sh logs`
2. Verify system requirements
3. Check Docker and Docker Compose versions
4. Review this documentation

🎯 Production Checklist

- ☐ Server meets minimum requirements
- ☐ Docker and Docker Compose installed
- ☐ Firewall configured
- ☐ SSL certificates installed (if using HTTPS)
- ☐ Domain configured (if using custom domain)
- ☐ Monitoring set up
- ☐ Backup strategy implemented
- ☐ Security updates scheduled

Note: This deployment guide assumes a Linux server. For Windows or macOS, use Docker Desktop and adjust paths accordingly.