Codelab Educare LMS
Production Deployment Guide

Codelab Educare LMS - Production Deployment Guide

Overview

This guide provides step-by-step instructions for deploying the Codelab Educare Learning Management System to production using Docker containers. The system is designed for the Nigerian education market with Paystack payment integration.

System Requirements

Minimum Server Specifications

- **CPU**: 2 vCPUs (4 vCPUs recommended)
- **RAM**: 4GB (8GB recommended)
- **Storage**: 50GB SSD (100GB recommended)
- **Network**: 10 Mbps upload/download
- **OS**: Ubuntu 20.04 LTS or higher

Software Dependencies

- Docker 20.10+
- Docker Compose 2.0+
- Git
- SSL certificate (Let's Encrypt recommended)

Pre-Deployment Checklist

1. Domain & SSL Setup

- [ ] Domain name purchased and configured
- [ ] DNS A record pointing to server IP
- [ ] SSL certificate obtained (Let's Encrypt recommended)
- [ ] Firewall configured (ports 80, 443, 22)

2. External Services Setup

- [ ] PostgreSQL database (managed service recommended)
- [ ] Paystack account with API keys
- [ ] Replit OAuth app configured
- [ ] Email service (optional but recommended)

3. Environment Configuration

- [ ] All environment variables configured
- [ ] Production secrets generated
- [ ] Database connection tested

Step-by-Step Deployment

Step 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/download/v2.20.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Add current user to docker group

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

Install Git

```
sudo apt install git -y
```

Step 2: Application Deployment

Clone the repository

```
git clone <your-repository-url> codelab-educare
cd codelab-educare
```

Create production environment file

```
cp .env.example .env.production
```

Edit environment variables (see configuration section below)

```
nano .env.production
```

Step 3: Environment Configuration

Edit `.env.production` with your production values:

Database Configuration (Use managed PostgreSQL service)

```
DATABASE_URL=postgresql://username:password@your-db-host:5432/codelab_educare
POSTGRES_DB=codelab_educare
POSTGRES_USER=your_username
POSTGRES_PASSWORD=your_secure_password
```

Session Configuration (Generate a strong secret)

```
SESSION_SECRET=your_very_secure_session_secret_minimum_32_characters
```

Paystack Configuration (Nigerian Payment Gateway)

PAYSTACK_SECRET_KEY=sk_live_your_paystack_secret_key
PAYSTACK_PUBLIC_KEY=pk_live_your_paystack_public_key

OAuth Configuration (Replit OAuth)

OAUTH_CLIENT_ID=your_replit_oauth_client_id
OAUTH_CLIENT_SECRET=your_replit_oauth_client_secret
OAUTH_REDIRECT_URI=https://yourdomain.com/auth/callback

Application Configuration

NODE_ENV=production
PORT=5000

Step 4: SSL Certificate Setup

Install Certbot for Let's Encrypt

sudo apt install snapd -y
sudo snap install --classic certbot

Create SSL directory

sudo mkdir -p /opt/codelab-educare/ssl

Generate SSL certificate

sudo certbot certonly --standalone -d yourdomain.com

Copy certificates to SSL directory

sudo cp /etc/letsencrypt/live/yourdomain.com/fullchain.pem /opt/codelab-educare/ssl/cert.pem
sudo cp /etc/letsencrypt/live/yourdomain.com/privkey.pem /opt/codelab-educare/ssl/key.pem

Set proper permissions

```
sudo chown -R $USER:$USER /opt/codelab-educare/ssl
chmod 600 /opt/codelab-educare/ssl/*
```

Step 5: Database Setup

Push database schema to production database

```
npm run db:push
```

Verify database connection

```
docker-compose exec app npm run check
```

Step 6: Application Launch

Build and start the application

```
docker-compose --env-file .env.production up -d
```

Check container status

```
docker-compose ps
```

View application logs

```
docker-compose logs -f app
```

Test health endpoint

```
curl http://localhost:5000/api/health
```

Step 7: Nginx Configuration

Update `nginx.conf` with your domain:

```
server {
    listen 443 ssl http2;
    server_name yourdomain.com;
```

```
    # SSL configuration
    ssl_certificate /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;
    # ... rest of configuration
}
```

Step 8: Domain & Firewall Configuration

Configure UFW firewall

```
sudo ufw enable
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
```

Test domain access

```
curl -I https://yourdomain.com/api/health
```

Production Monitoring

Health Checks

- **Application Health**: `https://yourdomain.com/api/health`
- **Database Connection**: Included in health check
- **SSL Certificate**: Monitor expiration dates

Log Monitoring

Application logs

```
docker-compose logs -f app
```

Nginx logs

```
docker-compose logs -f nginx
```

Database logs (if using local PostgreSQL)

```
docker-compose logs -f db
```

Performance Monitoring

Container resource usage

```
docker stats
```

System resource usage

```
htop
```

Database performance

```
docker-compose exec db psql -U postgres -d codelab_educare -c "SELECT * FROM pg_stat_activity;"
```

Backup Strategy

Database Backup

Daily database backup script

```
#!/bin/bash
DATE=$(date +%Y%m%d_%H%M%S)
docker-compose exec db pg_dump -U postgres codelab_educare > backup_${DATE}.sql
```

Keep last 7 days of backups

```
find /path/to/backups -name "backup_*.sql" -mtime +7 -delete
```

File Backup

Backup uploaded files

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

Backup configuration

tar -czf config_backup_$(date +%Y%m%d).tar.gz .env.production docker-compose.yml nginx.conf

SSL Certificate Renewal

Create renewal script

sudo crontab -e

Add this line for automatic renewal at 3 AM daily

0 3 * * * certbot renew --quiet && docker-compose restart nginx

Scaling Considerations

Horizontal Scaling

- Use a load balancer (nginx or cloud load balancer)
- Deploy multiple app containers
- Use Redis for session storage
- Implement CDN for static files

Vertical Scaling

- Increase server resources
- Optimize database connections
- Enable database connection pooling

Security Hardening

Application Security

- [ ] Environment variables properly secured
- [ ] File upload restrictions configured
- [ ] Rate limiting enabled in nginx
- [ ] Security headers configured
- [ ] Database access restricted

Server Security

- [ ] SSH key authentication only
- [ ] Fail2ban installed and configured
- [ ] Regular security updates

- [ ] Firewall properly configured
- [ ] SSL/TLS properly configured

Troubleshooting

Common Issues

Container Won't Start

Check container logs

docker-compose logs app

Check environment variables

docker-compose config

Verify database connection

docker-compose exec app node -e "console.log(process.env.DATABASE_URL)"

Database Connection Issues

Test database connectivity

docker-compose exec app npm run db:check

Check database logs

docker-compose logs db

Verify database credentials

psql "postgresql://username:password@host:port/database"

SSL Certificate Issues

Check certificate validity

openssl x509 -in /opt/codelab-educare/ssl/cert.pem -text -noout

Test SSL configuration

curl -I https://yourdomain.com

Check nginx configuration

docker-compose exec nginx nginx -t

Performance Issues

Monitor resource usage

docker stats

Check database performance

docker-compose exec db psql -U postgres -c "SELECT * FROM pg_stat_activity;"

Analyze slow queries

docker-compose exec db psql -U postgres -c "SELECT query, mean_time, calls FROM pg_stat_statements ORDER BY mean_time DESC LIMIT 10;"

Maintenance Tasks

Weekly Tasks

- [ ] Review application logs
- [ ] Check disk space usage
- [ ] Verify backups are working
- [ ] Update security patches

Monthly Tasks

- [ ] Review performance metrics
- [ ] Update dependencies
- [ ] Test disaster recovery
- [ ] Review security logs

Quarterly Tasks

- [ ] Full security audit
- [ ] Capacity planning review
- [ ] Disaster recovery testing
- [ ] Performance optimization

Support & Maintenance

Log Files Location

- Application logs: `docker-compose logs app`
- Nginx logs: `docker-compose logs nginx`
- Database logs: `docker-compose logs db`

Configuration Files

- Application: `.env.production`
- Docker: `docker-compose.yml`
- Nginx: `nginx.conf`

Key Commands

Restart application

docker-compose restart app

Update application

git pull && docker-compose up -d --build

Scale application

docker-compose up -d --scale app=3

View real-time logs

docker-compose logs -f

Clean up old images

```
docker system prune -a
```

Emergency Procedures

Application Rollback

Rollback to previous version

```
git checkout <previous-commit-hash>
docker-compose up -d --build
```

Database Recovery

Restore from backup

```
docker-compose exec db psql -U postgres -d codelab_educare <
backup_YYYYMMDD_HHMMSS.sql
```

Quick Health Check

```
#!/bin/bash
echo "=== System Health Check ==="
echo "Date: $(date)"
echo "Uptime: $(uptime)"
echo "Docker Status: $(docker --version)"
echo "App Status: $(curl -s http://localhost:5000/api/health | jq .status)"
echo "SSL Status: $(curl -I -s https://yourdomain.com | head -1)"
echo "Disk Usage: $(df -h / | tail -1)"
echo "Memory Usage: $(free -h)"
```

This deployment guide ensures a robust, secure, and scalable production deployment of the Codelab Educare LMS system.

Codelab Educare LMS - Production Deployment Guide