Course Platform DevOps Setup - Deployment Summary

Setup Complete!

Your comprehensive DevOps deployment setup for the Course Platform has been successfully created. This production-ready infrastructure supports multiple environments with automated deployments, security best practices, and monitoring.

What's Been Created

Docker Configuration

- **Multi-stage Dockerfile** optimized for Next.js production builds
- **Environment-specific Docker Compose files**:
- `docker-compose.dev.yml` - Development environment
- `docker-compose.staging.yml` - Staging environment
- `docker-compose.prod.yml` - Production with load balancing
- **Optimized .dockerignore** for smaller image sizes

Nginx Configuration

- **Reverse proxy setup** for multiple environments
- **SSL/TLS configuration** with security headers
- **Load balancing** for production environment
- **Rate limiting** and security protection
- **Domain routing**:
- `dev.example.com` → Development
- `staging.example.com` → Staging
- `www.example.com` → Production
- `www.example.com/dev` → Static test area

Deployment Scripts

- **`deploy.sh`** - Main deployment script with full options
- **`deploy_dev.sh`** - Quick development deployment
- **`deploy_staging.sh`** - Staging deployment with backup
- **`db_migrate.sh`** - Database migration management
- **`backup.sh`** - Automated backup creation
- **`rollback.sh`** - Rollback to previous versions
- **`server_init.sh`** - VPS server initialization

CI/CD Pipeline

- **GitHub Actions workflow** with:
- Automated testing (unit tests, linting, security scans)
- Docker image building and registry push

- Environment-specific deployments
- Health checks and notifications
- Performance testing with Lighthouse

VS Code Integration

- **Tasks configuration** for one-click deployments
- **Debug configurations** for Next.js and API routes
- **Recommended extensions** for optimal development
- **Settings optimization** for the project

Comprehensive Documentation

- **README.md** - Complete setup and usage guide
- **DOMAIN_SETUP.md** - DNS and domain configuration
- **SECURITY.md** - Security best practices and hardening
- **TROUBLESHOOTING.md** - Common issues and solutions

Security Features

- **SSL/TLS encryption** with Let's Encrypt integration
- **Security headers** and HSTS
- **Rate limiting** and DDoS protection
- **Firewall configuration** (UFW)
- **Fail2ban** intrusion prevention
- **Non-root containers** for security
- **Environment variable protection**

Quick Start Guide

1. Environment Setup

```
# Copy and configure environment files
cp app/.env.example app/.env.dev
cp app/.env.example app/.env.staging
cp app/.env.example app/.env.prod

# Edit with your configuration
nano app/.env.dev
nano app/.env.staging
nano app/.env.prod
```

2. Development Environment

```
# Start development environment
./scripts/deploy_dev.sh

# Or manually
docker compose -f docker-compose.dev.yml up -d

# Run migrations
./scripts/db_migrate.sh dev

# Access at: http://localhost:3000
```

3. Server Setup (Production)

```
# On your VPS server, run the initialization script
./scripts/server_init.sh

# Configure DNS records for your domains
# Generate SSL certificates
sudo certbot --nginx -d example.com -d www.example.com

# Deploy to production
./scripts/deploy.sh prod --backup
```

Environment URLs

Environment	URL	Purpose
Development	`http://localhost:3000`	Local development
Development	`http://dev.example.com`	Remote development
Staging	`https://staging.example.com`	Pre-production testing
Production	`https://www.example.com`	Live production site
Test Area	`https://www.example.com/dev`	Static test cases

Key Commands

Deployment

```
./scripts/deploy.sh dev                  # Deploy to development
./scripts/deploy.sh staging --backup     # Deploy to staging with backup
./scripts/deploy.sh prod --force         # Force deploy to production
```

Database Management

```
./scripts/db_migrate.sh dev deploy         # Run migrations
./scripts/db_migrate.sh dev status         # Check migration status
./scripts/db_migrate.sh dev seed           # Seed database
./scripts/db_migrate.sh dev studio         # Open Prisma Studio
```

Backup & Recovery

```
./scripts/backup.sh prod                   # Create production backup
./scripts/rollback.sh prod                 # Rollback to latest backup
./scripts/rollback.sh --list              # List available backups
```

Monitoring

```
docker compose -f docker-compose.dev.yml ps       # Check service status
docker compose -f docker-compose.dev.yml logs -f  # View logs
curl http://localhost:3000/health                 # Health check
```

Next Steps

1. Configure Your Environment

- [ ] Update environment files with real credentials
- [ ] Set up your domain DNS records
- [ ] Configure YooKassa payment credentials
- [ ] Set up monitoring and alerting

2. Security Setup

- [ ] Generate SSL certificates with Let’s Encrypt
- [ ] Configure firewall rules
- [ ] Set up SSH key authentication
- [ ] Review and update security settings

3. Production Deployment

- [ ] Run server initialization script on VPS
- [ ] Configure domain routing
- [ ] Deploy to staging for testing
- [ ] Deploy to production
- [ ] Set up monitoring and backups

4. CI/CD Setup

- [ ] Configure GitHub repository secrets
- [ ] Set up deployment keys
- [ ] Test automated deployments
- [ ] Configure notifications (Slack, email)

Security Checklist

- [ ] Environment variables secured
- [ ] SSL certificates configured
- [ ] Firewall rules applied
- [ ] Fail2ban configured
- [ ] Database access restricted
- [ ] Non-root containers used
- [ ] Security headers enabled
- [ ] Rate limiting configured
- [ ] Backup encryption enabled
- [ ] Monitoring alerts set up

Support & Resources

Documentation

- Complete Setup Guide (./docs/README.md)
- Domain Configuration (./docs/DOMAIN_SETUP.md)
- Security Guide (./docs/SECURITY.md)
- Troubleshooting (./docs/TROUBLESHOOTING.md)

Health Endpoints

- Development: `http://localhost:3000/health`
- Staging: `https://staging.example.com/health`
- Production: `https://www.example.com/health`

Log Locations

- Application: `/var/www/course-platform/logs/`
- Nginx: `/var/log/nginx/`
- System: `/var/log/syslog`

Congratulations!

You now have a production-ready DevOps setup with:

- Multi-environment support (dev, staging, prod)
- Automated deployments with rsync
- Docker containerization with optimization
- Nginx reverse proxy with SSL
- Database management and migrations
- Backup and rollback capabilities
- Security hardening and monitoring
- CI/CD pipeline with GitHub Actions
- VS Code integration for development
- Comprehensive documentation

Your Course Platform is ready for professional deployment!

Created: December 2024
Version: 1.0.0
Status: Production Ready