# Deployment Guide

This guide covers deploying the AI Command Center to various platforms.

## Quick Deploy Options

### Option 1: Vercel + Railway (Recommended)

#### 1. Deploy Database (Railway)

```
# Create account at railway.app
# Click "New Project" → "Provision PostgreSQL"
# Copy the connection string
```

#### 2. Deploy Application (Vercel)

```
# Install Vercel CLI
npm i -g vercel

# Deploy from project root
cd aicmdcenter
vercel

# Add environment variables in Vercel dashboard
# DATABASE_URL from Railway
# NEXTAUTH_SECRET (generate with: openssl rand -base64 32)
# NEXTAUTH_URL (your deployment URL)
```

#### 3. Run Database Migration

```
# After deployment, run migration
npx prisma migrate deploy
```

### Option 2: Fly.io (Full Stack)

#### 1. Install Fly CLI

```
# macOS
brew install flyctl

# Linux
curl -L https://fly.io/install.sh | sh
```

#### 2. Initialize and Deploy

```
# Login and create app
fly auth login
fly launch

# Add environment variables
fly secrets set DATABASE_URL="your-postgres-url"
fly secrets set NEXTAUTH_SECRET="your-secret"
fly secrets set NEXTAUTH_URL="https://your-app.fly.dev"

# Deploy
fly deploy
```

# Environment Variables

## Required for Production

```
# Database
DATABASE_URL="postgresql://user:pass@host:5432/dbname"

# NextAuth
NEXTAUTH_URL="https://your-domain.com"
NEXTAUTH_SECRET="your-super-secret-key"

# AI Services (Phase 2)
OPENAI_API_KEY="sk-your-real-openai-key"
```

## Platform APIs (Phase 2)

```
# Etsy
ETSY_CLIENT_ID="your-etsy-client-id"
ETSY_CLIENT_SECRET="your-etsy-client-secret"

# Gumroad
GUMROAD_ACCESS_TOKEN="your-gumroad-token"
```

# Docker Deployment

### 1. Build and Run

```
# Build image
docker build -t ai-command-center .

# Run with environment file
docker run --env-file .env -p 3000:3000 ai-command-center
```

### 2. Docker Compose

```yaml
# docker-compose.prod.yml
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=${DATABASE_URL}
      - NEXTAUTH_URL=${NEXTAUTH_URL}
      - NEXTAUTH_SECRET=${NEXTAUTH_SECRET}
    depends_on:
      - postgres

  postgres:
    image: postgres:15
    environment:
      POSTGRES_DB: aicmdcenter
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: password
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

volumes:
  postgres_data:
```

# Database Setup

## PostgreSQL Setup

```
# Local development
createdb aicmdcenter
npx prisma migrate dev

# Production
npx prisma migrate deploy
```

## Backup Strategy

```
# Backup
pg_dump $DATABASE_URL > backup.sql

# Restore
psql $DATABASE_URL < backup.sql
```

# SSL and Security

## SSL Certificate (Let’s Encrypt)

```
# Using Certbot
sudo certbot --nginx -d your-domain.com
```

### Security Headers (nginx)

```
# /etc/nginx/sites-available/ai-command-center
server {
    listen 443 ssl http2;
    server_name your-domain.com;

    ssl_certificate /path/to/cert.pem;
    ssl_certificate_key /path/to/private.key;

    # Security headers
    add_header X-Frame-Options DENY;
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";
    add_header Strict-Transport-Security "max-age=63072000" always;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        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;
        proxy_cache_bypass $http_upgrade;
    }
}
```

## Monitoring and Logging

### Health Check Endpoint

```
// pages/api/health.ts
export default function handler(req, res) {
  res.status(200).json({
    status: 'ok',
    timestamp: new Date().toISOString(),
    version: process.env.npm_package_version
  })
}
```

### Application Monitoring

```
# Install PM2 for process management
npm install -g pm2

# Start application
pm2 start npm --name "ai-command-center" -- start

# Monitor
pm2 monit

# Auto-restart on file changes
pm2 start ecosystem.config.js --env production
```

# Performance Optimization

## Next.js Optimization

```
// next.config.js
module.exports = {
  // Enable compression
  compress: true,

  // Image optimization
  images: {
    domains: ['your-cdn-domain.com'],
    formats: ['image/webp', 'image/avif'],
  },

  // Bundle analysis
  webpack: (config, { dev, isServer }) => {
    if (!dev && !isServer) {
      config.resolve.alias = {
        ...config.resolve.alias,
        react: 'preact/compat',
        'react-dom': 'preact/compat',
      }
    }
    return config
  }
}
```

## Database Optimization

```
-- Add indexes for common queries
CREATE INDEX idx_products_user_id ON "Product"("userId");
CREATE INDEX idx_products_status ON "Product"("status");
CREATE INDEX idx_agent_logs_created_at ON "AgentLog"("createdAt");
CREATE INDEX idx_revenues_date ON "Revenue"("date");
```

# Troubleshooting

## Common Issues

### Build Errors

```
# Clear Next.js cache
rm -rf .next

# Reinstall dependencies
rm -rf node_modules package-lock.json
npm install
```

### Database Connection Issues

```
# Test connection
npx prisma db pull

# Reset database (development only!)
npx prisma migrate reset
```

**Memory Issues**

```
# Increase Node.js memory limit
NODE_OPTIONS="--max-old-space-size=4096" npm run build
```

## Success Checklist

- [ ] Application builds successfully
- [ ] Database migrations run without errors
- [ ] All environment variables are set
- [ ] SSL certificate is configured
- [ ] Health check endpoint returns 200
- [ ] Authentication flow works
- [ ] Dashboard loads and displays data
- [ ] AI agents can be triggered
- [ ] No console errors in browser

Your AI Command Center is now ready for production! 🚀