🚀 Complete Deployment Guide

This guide covers deploying your multi-brand content platform to various hosting providers with full functionality including content automation, API integrations, and brand switching.

📋 Pre-Deployment Setup

1. Environment Configuration

Create your environment file:

```
# .env.production
# Core Settings
NEXT_PUBLIC_BRAND_TYPE=luxury-guide
NEXT_PUBLIC_SITE_URL=https://yourdomain.com
NEXTAUTH_URL=https://yourdomain.com
NEXTAUTH_SECRET=your-super-secret-key

# Database
DATABASE_URL=postgresql://username:password@host:port/database

# AI Content Generation
ABACUSAI_API_KEY=your-abacusai-key
OPENAI_API_KEY=your-openai-key

# Google Services
GOOGLE_ANALYTICS_ID=G-XXXXXXXXXX
GOOGLE_SEARCH_CONSOLE_KEY={"type":"service_account"...}
GOOGLE_TAG_MANAGER_ID=GTM-XXXXXXX

# Email & Notifications
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USERNAME=your-email@gmail.com
SMTP_PASSWORD=your-app-password

# Social Media (Optional)
INSTAGRAM_ACCESS_TOKEN=your-instagram-token
TIKTOK_ACCESS_TOKEN=your-tiktok-token

# AWS S3 (Optional)
AWS_ACCESS_KEY_ID=your-aws-key
AWS_SECRET_ACCESS_KEY=your-aws-secret
AWS_BUCKET_NAME=your-bucket-name

# Security
CRON_SECRET=your-cron-secret-for-automation
```

2. Database Setup

Option A: Neon (Recommended)

```
# 1. Create account at neon.tech
# 2. Create new project
# 3. Copy connection string
DATABASE_URL=postgresql://username:password@ep-xxx.us-east-1.aws.neon.tech/neondb
```

Option B: Railway

```
# 1. Create account at railway.app
# 2. Add PostgreSQL service
# 3. Copy DATABASE_URL from variables
```

Option C: Supabase

```
# 1. Create project at supabase.com
# 2. Get connection string from Settings > Database
DATABASE_URL=postgresql://postgres:password@db.xxx.supabase.co:5432/postgres
```

3. Build Preparation

```
# Install dependencies
yarn install

# Generate Prisma client
yarn prisma generate

# Run database migrations
yarn prisma migrate deploy

# Build the application
yarn build

# Test production build locally
yarn start
```

🌐 Platform-Specific Deployment

Vercel (Recommended)

Automatic Deployment

1. **Connect Repository**:
   - Go to vercel.com
   - Import your GitHub repository
   - Select project

2. **Configure Environment**:
   `bash`
   `# In Vercel dashboard > Settings > Environment Variables`
   `# Add all variables from .env.production`

3. **Database Integration**:
   - Add Neon integration from Vercel marketplace
   - Or manually add DATABASE_URL

4. **Custom Domain**:
   - Add domain in Project Settings > Domains
   - Configure DNS with your provider

5. **Deploy**:
   `bash`
   `# Automatic deployment on git push`
   `git push origin main`

Manual Deployment

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

# Login and deploy
vercel login
vercel --prod

# Set environment variables
vercel env add NEXT_PUBLIC_BRAND_TYPE
vercel env add DATABASE_URL
# ... add all environment variables

# Redeploy with environment
vercel --prod
```

Vercel Cron Jobs

Add `vercel.json`:

```
{
  "crons": [
    {
      "path": "/api/cron/auto-generate",
      "schedule": "0 */2 * * *"
    }
  ]
}
```

Netlify

1. **Build Settings**:
   ```bash
   # Build command
   yarn build

# Publish directory
out

# Add to netlify.toml
[build]
command = “yarn build”
publish = “out”
```

[[redirects]]
from = “/*”
to = “/index.html”
status = 200
```

1. **Environment Variables**:
   - Go to Site Settings > Environment Variables
   - Add all required variables

2. **Functions** (for cron jobs):
   ```javascript
   // netlify/functions/auto-generate.js
   const { schedule } = require(‘@netlify/functions’)

const handler = schedule(‘0 /2 * * ’, async (event) => {
// Call your auto-generation endpoint
const response = await fetch( `${process.env.URL}/api/cron/auto-generate` , {
method: ‘POST’,
headers: {
‘Authorization’: `Bearer ${process.env.CRON_SECRET}`
}
})

```
  return { statusCode: 200, body: JSON.stringify({ success: true }) }
```

})

module.exports.handler = handler
```

## Railway

1. **Deploy from GitHub**:
   - Connect GitHub repository
   - Select project to deploy

2. **Environment Variables**:
   `bash`
   `# In Railway dashboard > Variables`
   `# Add all environment variables`

3. **Database**:
   `bash`
   `# Add PostgreSQL plugin`
   `# Copy DATABASE_URL to variables`

4. **Custom Domain**:
   `bash`
   `# In Settings > Domains`
   `# Add your custom domain`
```

DigitalOcean App Platform

1. **Create App**:
   ```yaml
   # .do/app.yaml
   name: yalla-london
   services:
   - name: web
   source_dir: /
   github:
   repo: yourusername/yalla-london
   branch: main
   run_command: yarn start
   build_command: yarn build
   environment_slug: node-js
   instance_count: 1
   instance_size_slug: basic-xxs
   routes:
   ```
   - path: /
     envs:
   - key: NEXT_PUBLIC_BRAND_TYPE
     value: luxury-guide
   - key: DATABASE_URL
     value: ${database.DATABASE_URL}
     databases:
   - engine: PG
     name: database
     num_nodes: 1
     size: basic-xs
     version: “13”
     ```
2. **Deploy**:
   ```bash
   # Push to GitHub, auto-deploys
   git push origin main
   ```
   ```

Self-Hosted (VPS)

Prerequisites

```
# Install Node.js 18+
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Install PM2 for process management
npm install -g pm2

# Install Nginx
sudo apt update
sudo apt install nginx

# Install PostgreSQL
sudo apt install postgresql postgresql-contrib
```

Deployment Steps

```
# 1. Clone repository
git clone https://github.com/yourusername/yalla-london.git
cd yalla-london

# 2. Install dependencies
yarn install

# 3. Set up environment
cp .env.example .env.production
# Edit .env.production with your settings

# 4. Set up database
sudo -u postgres psql
CREATE DATABASE yallalondon;
CREATE USER yallalondon WITH PASSWORD 'yourpassword';
GRANT ALL PRIVILEGES ON DATABASE yallalondon TO yallalondon;
\q

# 5. Run migrations
yarn prisma migrate deploy

# 6. Build application
yarn build

# 7. Start with PM2
pm2 start ecosystem.config.js

# 8. Configure Nginx
sudo nano /etc/nginx/sites-available/yallalondon
```

Nginx Configuration

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

    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;
    }
}
```

PM2 Configuration

```
// ecosystem.config.js
module.exports = {
  apps: [{
    name: 'yalla-london',
    script: 'yarn',
    args: 'start',
    cwd: '/path/to/yalla-london',
    env: {
      NODE_ENV: 'production'
    },
    instances: 'max',
    exec_mode: 'cluster'
  }]
}
```

SSL Certificate

```
# Install Certbot
sudo apt install certbot python3-certbot-nginx

# Get SSL certificate
sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com

# Auto-renewal
sudo crontab -e
# Add: 0 12 * * * /usr/bin/certbot renew --quiet
```

Cron Jobs (Self-Hosted)

```
# Edit crontab
crontab -e

# Add content automation (every 2 hours)
0 */2 * * * curl -X POST -H "Authorization: Bearer your-cron-secret" https://yourdomain.com/api/cron/auto-generate

# Add database cleanup (daily)
0 2 * * * curl -X POST -H "Authorization: Bearer your-cron-secret" https://yourdomain.com/api/cron/cleanup
```

🔧 Post-Deployment Configuration

1. Admin Panel Setup

1. **Access admin panel**: `https://yourdomain.com/admin`
2. **Configure API keys** in Settings tab
3. **Test integrations** using the test buttons
4. **Set up content automation rules**

2. Content Generation

```
# Test content generation
curl -X POST https://yourdomain.com/api/content/auto-generate \
  -H "Content-Type: application/json" \
  -d '{
    "type": "blog_post",
    "category": "london-guide",
    "language": "en"
  }'

# Test automation
curl -X POST https://yourdomain.com/api/cron/auto-generate \
  -H "Authorization: Bearer your-cron-secret"
```

3. SEO Setup

1. **Google Search Console**:
   - Add property for your domain
   - Upload service account key in admin panel
   - Verify ownership

2. **Google Analytics**:
   - Create GA4 property
   - Add measurement ID to admin panel

3. **Submit Sitemap**:
   `https://yourdomain.com/sitemap.xml`

4. Social Media Integration

1. **Instagram Business**:
   - Get access token from Facebook Developer Console
   - Add to admin panel settings

2. **TikTok for Business**:
   - Apply for TikTok Marketing API
   - Configure access token

🔄 Multi-Brand Deployments

Strategy 1: Separate Deployments

```
# Brand 1: Yalla London
NEXT_PUBLIC_BRAND_TYPE=luxury-guide
NEXT_PUBLIC_SITE_URL=https://yallalondon.com

# Brand 2: Dubai Fine Dining
NEXT_PUBLIC_BRAND_TYPE=restaurant-guide
NEXT_PUBLIC_SITE_URL=https://dubaifinedining.com

# Brand 3: London Kids Guide
NEXT_PUBLIC_BRAND_TYPE=kids-retail
NEXT_PUBLIC_SITE_URL=https://londonkidsguide.com
```

Strategy 2: Branch-Based Deployment

```
# Main branch: Default brand
git checkout main
# Deploy to main domain

# Brand branches
git checkout -b dubai-restaurants
# Update .env with restaurant-guide brand
# Deploy to subdomain or separate domain
```

Strategy 3: Environment-Based

```
# Vercel: Multiple environments
vercel env add NEXT_PUBLIC_BRAND_TYPE luxury-guide --environment production
vercel env add NEXT_PUBLIC_BRAND_TYPE restaurant-guide --environment preview

# Deploy different brands to different aliases
vercel --prod --target production
vercel --prod --target preview
```

📊 Monitoring & Analytics

Application Monitoring

```
# Add monitoring tools
yarn add @vercel/analytics @vercel/speed-insights

# Environment variables for monitoring
NEXT_PUBLIC_VERCEL_ANALYTICS=1
NEXT_PUBLIC_SPEED_INSIGHTS=1
```

Error Tracking

```
# Add Sentry
yarn add @sentry/nextjs

# Configure in next.config.js
const { withSentryConfig } = require('@sentry/nextjs');

module.exports = withSentryConfig({
  // Next.js config
}, {
  // Sentry config
  silent: true,
  org: 'your-org',
  project: 'yalla-london'
});
```

Performance Monitoring

```
# Lighthouse CI
npm install -g @lhci/cli

# Configure .lighthouserc.js
module.exports = {
  ci: {
    collect: {
      url: ['https://yourdomain.com'],
      startServerCommand: 'yarn start'
    },
    assert: {
      assertions: {
        'categories:performance': ['warn', { minScore: 0.9 }],
        'categories:accessibility': ['error', { minScore: 0.9 }],
        'categories:seo': ['error', { minScore: 0.9 }]
      }
    }
  }
};
```

🔒 Security Configuration

Content Security Policy

```
// next.config.js
const nextConfig = {
  async headers() {
    return [
      {
        source: '/(.*)',
        headers: [
          {
            key: 'Content-Security-Policy',
            value: `
              default-src 'self';
              script-src 'self' 'unsafe-eval' 'unsafe-inline' https://www.google-ana-
lytics.com;
              style-src 'self' 'unsafe-inline';
              img-src 'self' data: https:;
              font-src 'self' data:;
              connect-src 'self' https://api.openai.com https://apps.abacus.ai;
            `.replace(/\s{2,}/g, ' ').trim()
          }
        ]
      }
    ]
  }
}
```

Rate Limiting

```
// middleware.ts
import { NextRequest, NextResponse } from 'next/server'
import { Ratelimit } from '@upstash/ratelimit'
import { Redis } from '@upstash/redis'

const ratelimit = new Ratelimit({
  redis: Redis.fromEnv(),
  limiter: Ratelimit.slidingWindow(10, '10 s'),
})

export async function middleware(request: NextRequest) {
  const ip = request.ip ?? '127.0.0.1'
  const { success } = await ratelimit.limit(ip)

  if (!success) {
    return NextResponse.json(
      { error: 'Too many requests' },
      { status: 429 }
    )
  }

  return NextResponse.next()
}
```

🚨 Troubleshooting

Common Issues

1. **Build Fails**:
   ```bash
   # Clear cache and rebuild
   rm -rf .next
   yarn build
   ```

2. **Database Connection**:
   ```bash
   # Test connection
   yarn prisma db push --accept-data-loss
   ```

3. **Environment Variables Not Loading**:
   ```bash
   # Check environment
   echo $NEXT_PUBLIC_BRAND_TYPE

   # Verify in code
   console.log(process.env.NEXT_PUBLIC_BRAND_TYPE)
   ```

1. **API Routes Failing**:
   ```bash
   # Check logs
   vercel logs
   # Or PM2 logs for self-hosted
   pm2 logs yalla-london
   ```

Performance Issues

```bash
# Analyze bundle size
yarn build --analyze

# Check Core Web Vitals
yarn lighthouse https://yourdomain.com

# Monitor API response times
curl -w "@curl-format.txt" -s -o /dev/null https://yourdomain.com/api/health
```

✅ Deployment Checklist

Pre-Launch

- [ ] Environment variables configured
- [ ] Database migrations applied
- [ ] Build completes successfully
- [ ] All API integrations tested
- [ ] Brand configuration verified
- [ ] SSL certificate installed
- [ ] Custom domain configured

Post-Launch

- [ ] Google Analytics tracking verified
- [ ] Search Console connected and verified
- [ ] Sitemap submitted
- [ ] Social media profiles linked
- [ ] Content generation working
- [ ] Automation rules configured
- [ ] Admin panel accessible
- [ ] Error monitoring active
- [ ] Performance metrics baseline established

Ongoing Maintenance

- [ ] Regular database backups
- [ ] Security updates applied
- [ ] Performance monitoring
- [ ] Content quality review
- [ ] SEO performance tracking
- [ ] Social media engagement
- [ ] User feedback collection

Your multi-brand content platform is now ready for deployment! 🎉