TreeHub Deployment Guide

This guide covers deploying TreeHub to various platforms with detailed configuration instructions.

🚀 Vercel Deployment (Recommended)

Vercel provides the optimal deployment experience for Next.js applications with zero-configuration setup.

Prerequisites

- GitHub, GitLab, or Bitbucket account
- Vercel account (free tier available)
- Production database (PostgreSQL)

Step-by-Step Deployment

1. Prepare Your Repository

```
# Ensure your code is committed and pushed
git add .
git commit -m "Prepare for deployment"
git push origin main
```

2. Connect to Vercel

1. Visit vercel.com (https://vercel.com) and sign in
2. Click “New Project”
3. Import your TreeHub repository
4. Vercel will automatically detect Next.js configuration

3. Configure Build Settings

Vercel automatically detects the correct settings:
- **Framework Preset**: Next.js
- **Build Command**: `cd app && npm run build`
- **Output Directory**: `app/.next`
- **Install Command**: `cd app && npm install --legacy-peer-deps`

4. Environment Variables

Add the following environment variables in Vercel dashboard:

Required Variables:

```
DATABASE_URL=postgresql://user:pass@host:5432/treehub
NEXTAUTH_SECRET=your-secret-key
NEXTAUTH_URL=https://your-domain.vercel.app
```

Optional but Recommended:

```
GOOGLE_MAPS_API_KEY=your-api-key
STRIPE_SECRET_KEY=your-stripe-key
SMTP_HOST=your-smtp-host
SMTP_USER=your-email
SMTP_PASS=your-password
AWS_ACCESS_KEY_ID=your-aws-key
AWS_SECRET_ACCESS_KEY=your-aws-secret
AWS_S3_BUCKET=your-bucket-name
```

5. Database Setup

Ensure your production database is ready:

```
# Run migrations
npx prisma migrate deploy

# Generate Prisma client
npx prisma generate
```

6. Deploy

Click “Deploy” in Vercel dashboard. The deployment process will:
1. Install dependencies
2. Build the application
3. Deploy to global edge network
4. Provide a live URL

7. Custom Domain (Optional)

1. Go to Project Settings → Domains
2. Add your custom domain
3. Configure DNS records as instructed
4. SSL certificate is automatically provisioned

Vercel Configuration File

The included `vercel.json` provides optimized settings:

```
{
  "framework": "nextjs",
  "buildCommand": "cd app && npm run build",
  "outputDirectory": "app/.next",
  "installCommand": "cd app && npm install --legacy-peer-deps",
  "functions": {
    "app/api/**/*.ts": {
      "maxDuration": 30
    }
  },
  "headers": [
    {
      "source": "/(.*)",
      "headers": [
        {
          "key": "X-Content-Type-Options",
          "value": "nosniff"
        }
      ]
    }
  ]
}
```

🐳 Docker Deployment

Dockerfile

Create a `Dockerfile` in the project root:

```
FROM node:18-alpine AS base

# Install dependencies only when needed
FROM base AS deps
RUN apk add --no-cache libc6-compat
WORKDIR /app

# Install dependencies
COPY app/package*.json ./
RUN npm ci --only=production --legacy-peer-deps

# Rebuild the source code only when needed
FROM base AS builder
WORKDIR /app
COPY app/package*.json ./
RUN npm ci --legacy-peer-deps

COPY app/ .

# Generate Prisma client
RUN npx prisma generate

# Build application
RUN npm run build

# Production image
FROM base AS runner
WORKDIR /app

ENV NODE_ENV production

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public

# Set the correct permission for prerender cache
RUN mkdir .next
RUN chown nextjs:nodejs .next

# Automatically leverage output traces to reduce image size
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ./.next/static

USER nextjs

EXPOSE 3000

ENV PORT 3000
ENV HOSTNAME "0.0.0.0"

CMD ["node", "server.js"]
```

Docker Compose

Create `docker-compose.yml`:

```yaml
version: '3.8'

services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=postgresql://postgres:password@db:5432/treehub
      - NEXTAUTH_SECRET=your-secret
      - NEXTAUTH_URL=http://localhost:3000
    depends_on:
      - db
    volumes:
      - ./app/.env.local:/app/.env.local

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

volumes:
  postgres_data:
```

Build and Run

```bash
# Build and start services
docker-compose up --build

# Run in background
docker-compose up -d

# View logs
docker-compose logs -f app

# Stop services
docker-compose down
```

☁️ AWS Deployment

Using AWS App Runner

1. **Create App Runner Service**:
   - Connect to your GitHub repository
   - Configure build settings
   - Set environment variables

2. **Build Configuration** (`apprunner.yaml`):

```yaml
version: 1.0
runtime: nodejs18
build:
  commands:
    build:
      - cd app
      - npm install --legacy-peer-deps
      - npx prisma generate
      - npm run build
run:
  runtime-version: 18
  command: cd app && npm start
  network:
    port: 3000
    env: PORT
  env:
    - name: NODE_ENV
      value: production
```

Using Elastic Beanstalk

1. **Prepare Application**:

```bash
# Create deployment package
cd app
npm run build
zip -r ../treehub-app.zip . -x "node_modules/*" ".next/*"
```

1. **Deploy to Beanstalk**:
   - Create new application
   - Upload deployment package
   - Configure environment variables
   - Set up RDS database

🌐 Other Platforms

Netlify

1. Connect repository to Netlify
2. Set build command: `cd app && npm run build`
3. Set publish directory: `app/.next`
4. Configure environment variables
5. Deploy

Railway

1. Connect GitHub repository
2. Railway auto-detects Next.js
3. Add environment variables
4. Deploy automatically

DigitalOcean App Platform

1. Create new app from GitHub
2. Configure build settings

3. Add database component
4. Set environment variables
5. Deploy

🔧 Environment-Specific Configurations

Development

```
NODE_ENV=development
NEXT_PUBLIC_APP_URL=http://localhost:3000
DATABASE_URL=postgresql://localhost:5432/treehub_dev
```

Staging

```
NODE_ENV=production
NEXT_PUBLIC_APP_URL=https://staging.treehubusa.com
DATABASE_URL=postgresql://staging-db-url
```

Production

```
NODE_ENV=production
NEXT_PUBLIC_APP_URL=https://treehubusa.com
DATABASE_URL=postgresql://production-db-url
```

📊 Performance Optimization

Build Optimization

- Enable output file tracing in `next.config.js`
- Use `standalone` output for smaller Docker images
- Implement proper caching strategies

Database Optimization

- Use connection pooling
- Implement read replicas for scaling
- Set up proper indexes

CDN Configuration

- Configure static asset caching
- Use image optimization services
- Implement proper cache headers

🔒 Security Considerations

Environment Variables

- Never commit `.env` files
- Use secure secret management
- Rotate secrets regularly

Database Security

- Use SSL connections
- Implement proper access controls
- Regular security updates

Application Security

- Enable HTTPS everywhere
- Implement proper CORS policies
- Use security headers

📈 Monitoring and Logging

Application Monitoring

- Set up error tracking (Sentry)
- Implement performance monitoring
- Configure uptime monitoring

Database Monitoring

- Monitor connection pools
- Track query performance
- Set up backup monitoring

Infrastructure Monitoring

- Monitor resource usage
- Set up alerting
- Track deployment metrics

🚨 Troubleshooting

Common Issues

Build Failures:
- Check Node.js version compatibility
- Verify environment variables
- Review dependency conflicts

Database Connection Issues:
- Verify connection string format
- Check network connectivity
- Confirm database credentials

Performance Issues:
- Review bundle size
- Check database query performance
- Analyze network requests

Debug Commands

```
# Check build output
npm run build

# Analyze bundle
npm run analyze

# Test database connection
npx prisma db pull

# Check environment variables
npm run env:check
```

📞 Support

For deployment issues:
- Check the troubleshooting section
- Review platform-specific documentation
- Contact support at deploy@treehubusa.com

Happy Deploying! 🚀