Deployment Guide

Overview

This guide covers deployment strategies for the Berean Bible Reading Plan Application across different environments and platforms.

Prerequisites

- Node.js 18+ installed
- Git repository access
- Environment variables configured
- Domain name (for production)

Environment Configuration

Environment Variables

Create appropriate `.env` files for each environment:

Development (.env.local)

```
# API Keys
ESV_API_KEY=your_development_esv_api_key

# Application URLs
NEXT_PUBLIC_APP_URL=http://localhost:3000

# Environment
NODE_ENV=development

# Debug flags
NEXT_PUBLIC_DEBUG=true
```

Production (.env.production)

```
# API Keys
ESV_API_KEY=your_production_esv_api_key

# Application URLs
NEXT_PUBLIC_APP_URL=https://your-domain.com

# Environment
NODE_ENV=production

# Security
NEXT_PUBLIC_DEBUG=false
```

Deployment Platforms

1. Vercel (Recommended)

Vercel provides seamless Next.js deployment with automatic builds and deployments.

Setup Steps

1. **Connect Repository**
   ```bash
   # Install Vercel CLI
   npm i -g vercel

# Login and deploy
vercel login
vercel
```

1. **Configure Environment Variables**
   - Go to Vercel Dashboard → Project → Settings → Environment Variables
   - Add all required environment variables
   - Set different values for Preview and Production

2. **Custom Domain**
   - Add your domain in Vercel Dashboard
   - Configure DNS records as instructed
   - SSL certificates are automatically managed

Vercel Configuration (vercel.json)

```json
{
  "framework": "nextjs",
  "buildCommand": "npm run build",
  "devCommand": "npm run dev",
  "installCommand": "npm install",
  "regions": ["iad1"],
  "env": {
    "ESV_API_KEY": "@esv-api-key"
  }
}
```

2. Netlify

Alternative platform with similar features to Vercel.

Setup Steps

1. **Connect Repository**
   - Link GitHub repository in Netlify dashboard
   - Configure build settings

2. **Build Configuration**
   ```toml
   # netlify.toml
   [build]
   command = "npm run build"
   publish = ".next"
   ```

```
[build.environment]
NODE_VERSION = “18”

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

3. Docker Deployment

For containerized deployments on any platform.

Dockerfile

```dockerfile
# Build stage
FROM node:18-alpine AS builder

WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production

COPY . .
RUN npm run build

# Production stage
FROM node:18-alpine AS runner

WORKDIR /app

# Create non-root user
RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

# Copy built application
COPY --from=builder /app/public ./public
COPY --from=builder /app/.next/standalone ./
COPY --from=builder /app/.next/static ./.next/static

USER nextjs

EXPOSE 3000

ENV PORT 3000
ENV HOSTNAME "0.0.0.0"

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

Docker Compose

```yaml
version: '3.8'

services:
  berean-bible-app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - ESV_API_KEY=${ESV_API_KEY}
      - NEXT_PUBLIC_APP_URL=${NEXT_PUBLIC_APP_URL}
    restart: unless-stopped

  nginx:
    image: nginx:alpine
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
      - ./ssl:/etc/nginx/ssl
    depends_on:
      - berean-bible-app
    restart: unless-stopped
```

4. Traditional VPS/Server

For deployment on virtual private servers or dedicated servers.

Setup Script

```bash
#!/bin/bash

# Update system
sudo apt update && sudo apt upgrade -y

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

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

# Clone and setup application
git clone <repository-url> /var/www/berean-bible-app
cd /var/www/berean-bible-app

# Install dependencies and build
npm ci
npm run build

# Configure PM2
pm2 start ecosystem.config.js
pm2 save
pm2 startup
```

PM2 Configuration (ecosystem.config.js)

```
module.exports = {
  apps: [{
    name: 'berean-bible-app',
    script: 'npm',
    args: 'start',
    cwd: '/var/www/berean-bible-app',
    instances: 'max',
    exec_mode: 'cluster',
    env: {
      NODE_ENV: 'production',
      PORT: 3000
    },
    error_file: './logs/err.log',
    out_file: './logs/out.log',
    log_file: './logs/combined.log',
    time: true
  }]
};
```

CI/CD Pipeline

GitHub Actions

Workflow Configuration (.github/workflows/deploy.yml)

```yaml
name: Deploy to Production

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
          cache: 'npm'

      - run: npm ci
      - run: npm run lint
      - run: npm run test
      - run: npm run build

  deploy:
    needs: test
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'

    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
          cache: 'npm'

      - run: npm ci
      - run: npm run build

      - name: Deploy to Vercel
        uses: amondnet/vercel-action@v20
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-org-id: ${{ secrets.ORG_ID }}
          vercel-project-id: ${{ secrets.PROJECT_ID }}
          vercel-args: '--prod'
```

Performance Optimization

Build Optimization

1. **Bundle Analysis**
   `bash`
   `  npm install --save-dev @next/bundle-analyzer`

2. **Image Optimization**
   - Use Next.js Image component
   - Configure image domains in next.config.js
   - Implement lazy loading

3. **Code Splitting**
   - Dynamic imports for large components
   - Route-based code splitting (automatic in Next.js)

Caching Strategy

1. **Static Assets**
   - Long-term caching for images, fonts
   - Versioned asset URLs

2. **API Responses**
   - Cache Bible passages locally
   - Implement service worker for offline access

3. **CDN Configuration**
   - Use Vercel Edge Network or CloudFlare
   - Geographic distribution

Monitoring & Analytics

Error Tracking

```
# Install Sentry for error tracking
npm install @sentry/nextjs
```

Performance Monitoring

- Core Web Vitals tracking
- Real User Monitoring (RUM)
- Server-side performance metrics

Analytics

- Google Analytics 4 integration
- Privacy-compliant user tracking
- Reading engagement metrics

Security Considerations

Environment Security

- Never commit API keys to repository
- Use environment variable management
- Rotate API keys regularly

Application Security

- Content Security Policy (CSP)
- HTTPS enforcement

- Input validation and sanitization

Infrastructure Security

- Regular security updates
- Firewall configuration
- SSL/TLS certificates

Backup & Recovery

Database Backup

- User progress data backup (if implemented)
- Configuration backup

Code Backup

- Git repository mirroring
- Automated backups to multiple locations

Disaster Recovery

- Deployment rollback procedures
- Emergency contact procedures
- Service restoration timeline

Troubleshooting

Common Issues

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

2. **API Issues**
   - Validate API keys
   - Check rate limiting
   - Monitor external service status

3. **Performance Issues**
   - Analyze bundle size
   - Check for memory leaks
   - Monitor server resources

Debug Commands

```
# Check build output
npm run build -- --debug

# Analyze bundle
npm run analyze

# Check dependencies
npm audit

# Performance profiling
npm run dev -- --profile
```

For specific deployment questions or issues, refer to the platform-specific documentation or contact the development team.