🚀 Deployment Guide

Complete deployment guide for the Strapi Template monorepo across multiple platforms.

📋 Table of Contents

🎯 Quick Deploy

One-Click Deploy Buttons

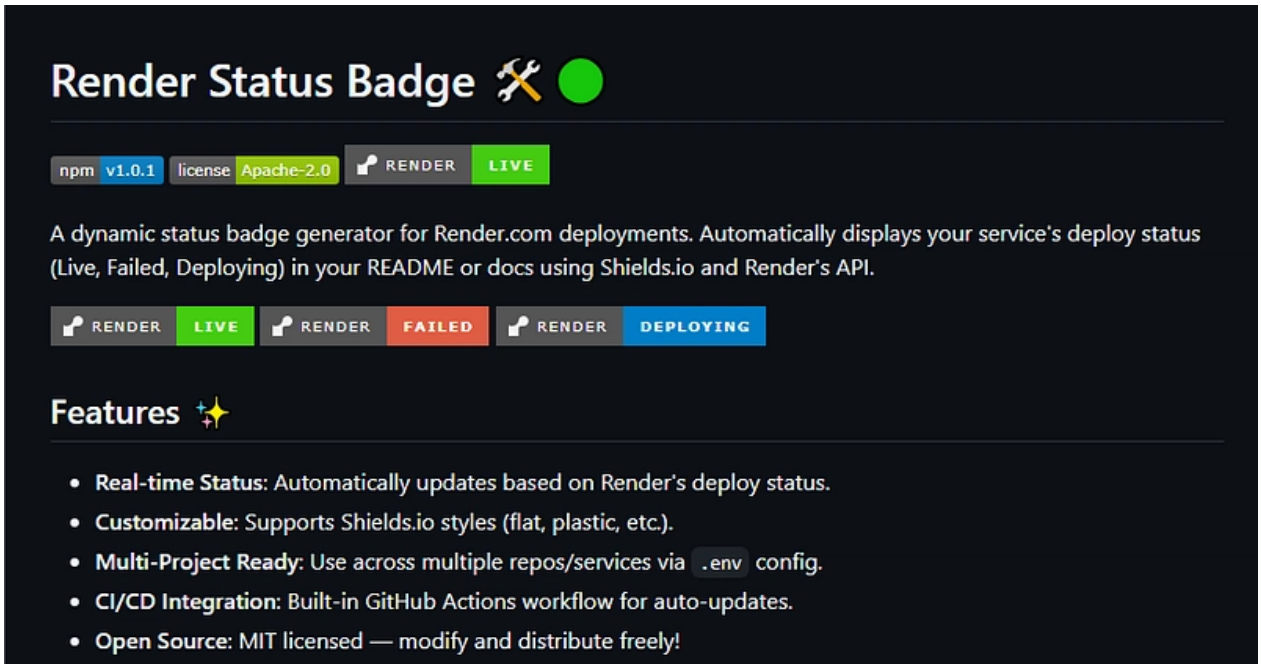
Google Cloud Run

Run on Google Cloud

(https://deploy.cloud.run?git_repo=https://github.com/executiveusa/strapi-template-new-world-kids.git)

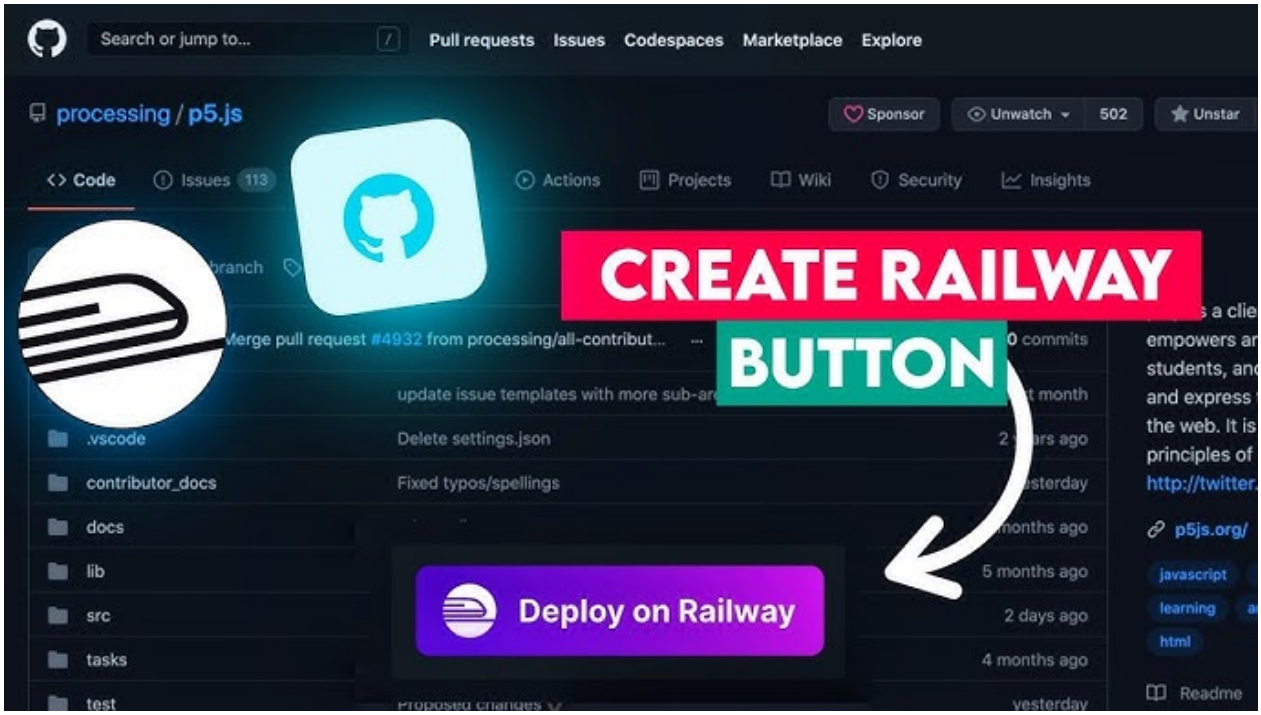
Render

[



Railway

[



☁️ Google Cloud Platform

Prerequisites

- Google Cloud SDK installed (Install Guide (https://cloud.google.com/sdk/docs/install))
- GCP Project created
- Billing enabled

Option 1: Automated Deployment Script

```
# Make script executable
chmod +x gcp-deploy.sh

# Run deployment
./gcp-deploy.sh
```

The script will guide you through:
1. Cloud Run deployment (Recommended)
2. App Engine deployment
3. Build only

Option 2: Cloud Run Manual Deployment

```
# Set your project
gcloud config set project YOUR_PROJECT_ID

# Build and deploy
gcloud builds submit --config cloudbuild.yaml

# Your app will be deployed automatically to Cloud Run
```

Option 3: App Engine Deployment

```
# Deploy to App Engine
gcloud app deploy app.yaml
```

Cloud Run Configuration

Update `cloudrun.yaml` with your project ID:

```
containers:
- image: gcr.io/YOUR_PROJECT_ID/strapi-template:latest
```

Then deploy:

```
gcloud run services replace cloudrun.yaml --region=us-west1
```

Environment Variables for GCP

Set environment variables in Cloud Console or via CLI:

```
gcloud run services update strapi-template \
  --update-env-vars="NODE_ENV=production,DATABASE_URL=your_db_url" \
  --region=us-west1
```

🎨 Render

Current Deployment

- **Service ID**: srv-d3vida0gjchc73d9mj8g
- **URL**: https://strapi-template-new-world-kids.onrender.com
- **Deploy Hook**: https://api.render.com/deploy/srv-d3vida0gjchc73d9mj8g?key=zSsWyMmdX7U

Manual Deployment

```
# Trigger deployment via hook
curl -X POST https://api.render.com/deploy/srv-d3vida0gjchc73d9mj8g?key=zSsWyMmdX7U
```

Via Dashboard

1. Go to Render Dashboard (https://dashboard.render.com/)

2. Select your service
3. Click “Manual Deploy” → “Deploy latest commit”

Configuration

The project includes `render.yaml` with:
- Node.js 22.11.0
- Yarn 1.22.19
- Auto-deploy on git push
- Health checks enabled

🐳 Docker

Local Development

```
# Build the image
docker build -t strapi-template .

# Run the container
docker run -p 1337:1337 \
  -e NODE_ENV=production \
  -e DATABASE_URL=your_db_url \
  strapi-template
```

Docker Compose

For local development with all services:

```
# Start all services
docker-compose up

# Start specific service
docker-compose up stellar-agents

# Build and start
docker-compose up --build

# Stop all services
docker-compose down
```

Multi-Stage Build

The Dockerfile uses multi-stage builds for optimization:
- **base**: Node.js 22 Alpine
- **dependencies**: Install all dependencies
- **builder**: Build the application
- **runner**: Production runtime (minimal size)

🔐 Environment Variables

Required Variables

```
# Node Environment
NODE_ENV=production
PORT=1337

# Database (PostgreSQL recommended for production)
DATABASE_CLIENT=postgres
DATABASE_HOST=your-db-host
DATABASE_PORT=5432
DATABASE_NAME=strapi
DATABASE_USERNAME=your-username
DATABASE_PASSWORD=your-password
DATABASE_SSL=true

# Strapi Admin
ADMIN_JWT_SECRET=your-admin-jwt-secret
JWT_SECRET=your-jwt-secret
API_TOKEN_SALT=your-api-token-salt
APP_KEYS=your-app-keys

# AI Services (if using stellar-agents)
OPENAI_API_KEY=your-openai-key
ANTHROPIC_API_KEY=your-anthropic-key
GOOGLE_API_KEY=your-google-key

# Supabase (if using)
SUPABASE_URL=your-supabase-url
SUPABASE_SERVICE_ROLE_KEY=your-supabase-key
```

Generate Secrets

```
# Generate random secrets
node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"
```

🏥 Health Checks

Endpoints

- **Strapi CMS**: `GET /admin` (responds with 200 when ready)
- **Stellar Agents**: `GET /health` (custom health endpoint)
- **Stream Service**: `GET /health`

Docker Health Check

Built into Dockerfile:

```
HEALTHCHECK --interval=30s --timeout=3s --start-period=40s --retries=3 \
    CMD node -e "require('http').get('http://localhost:${PORT:-1337}/admin', (r) =>
{process.exit(r.statusCode === 200 ? 0 : 1)})"
```

Google Cloud Health Checks

Configured in `cloudrun.yaml`:
- **Startup Probe**: 10s initial delay, 10s period
- **Liveness Probe**: 30s initial delay, 30s period
- **Readiness Probe**: 10s initial delay, 10s period

🔧 Troubleshooting

Build Errors

Issue: TypeScript compilation errors

```
# Clear cache and rebuild
rm -rf node_modules .turbo dist
yarn install
yarn build
```

Issue: Node version mismatch

```
# Use correct Node version
nvm use 22
# or
nvm install 22
```

Docker Issues

Issue: Build fails due to memory

```
# Increase Docker memory limit (Docker Desktop settings)
# Or build with more memory:
docker build --memory=4g -t strapi-template .
```

Issue: Container exits immediately

```
# Check logs
docker logs <container-id>

# Run interactively
docker run -it strapi-template sh
```

Deployment Issues

Issue: 502 Bad Gateway
- Check health endpoint is responding
- Verify PORT environment variable matches exposed port
- Check application logs

Issue: Database connection errors
- Verify database credentials
- Check database is accessible from deployment environment
- For Cloud SQL, ensure Cloud SQL Proxy is configured

📊 Performance Optimization

Production Best Practices

1. **Enable Gzip compression** (handled by reverse proxy)
2. **Use CDN** for static assets
3. **Database Connection Pooling**
4. **Redis Caching** (optional)
5. **Monitor Memory Usage**

Scaling

Cloud Run

- Automatically scales 1-10 instances
- Configure in `cloudrun.yaml`

Render

- Auto-scaling based on load
- Configure in dashboard

Docker Compose

```
# Scale specific service
docker-compose up --scale stellar-agents=3
```

📈 Monitoring

Google Cloud

```
# View logs
gcloud run services logs read strapi-template --region=us-west1

# Monitor metrics
gcloud monitoring dashboards list
```

Render

View logs in dashboard:
- Real-time logs
- Historical logs (last 7 days)
- Metrics dashboard

🔄 CI/CD

Google Cloud Build

Triggers automatically on git push (after connecting repository):

```
# Manual trigger
gcloud builds submit --config cloudbuild.yaml
```

GitHub Actions

(Optional) Add `.github/workflows/deploy.yml`:

```yaml
name: Deploy to Cloud Run
on:
  push:
    branches: [main]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: google-github-actions/setup-gcloud@v0
      - run: gcloud builds submit --config cloudbuild.yaml
```

📞 Support

- **Render Issues**: Render Support (https://render.com/docs)
- **GCP Issues**: Google Cloud Support (https://cloud.google.com/support)
- **Docker Issues**: Docker Documentation (https://docs.docker.com/)

🎉 Success!

Your application should now be deployed and running. Access it at your deployment URL and verify all services are healthy.

For production deployments, remember to:
- [ ] Set all required environment variables
- [ ] Configure SSL/TLS certificates
- [ ] Set up monitoring and alerts
- [ ] Configure backups
- [ ] Review security settings