Deployment Guide

Complete guide for deploying the 3I/ATLAS Flight Tracker to production.

Table of Contents

Pre-Deployment Checklist

Before deploying, ensure:

- [ ] Trajectory data generated (`python3 generate_atlas_trajectory.py`)
- [ ] All dependencies installed (`npm install`)
- [ ] Production build successful (`npm run build`)
- [ ] Assets optimized (images, data files)
- [ ] Environment variables configured
- [ ] Error boundaries in place
- [ ] Analytics integrated (optional)
- [ ] Performance tested locally

Static Hosting

The built application is a static site that can be hosted anywhere.

Build Steps

```
cd frontend
npm install
npm run build
```

Output directory: `frontend/dist/`

Required Files Structure

```
dist/
├── index.html
├── assets/
│   ├── index-[hash].js
│   ├── index-[hash].css
│   └── ...
└── data/
    ├── trajectory_static.json
    └── timeline_events.json
```

Vercel Deployment

Option 1: CLI Deployment

```bash
# Install Vercel CLI
npm install -g vercel

# Navigate to frontend
cd frontend

# Deploy
vercel

# Production deployment
vercel --prod
```

Option 2: GitHub Integration

1. **Push to GitHub:**

```bash
git add .
git commit -m "Deploy 3I/ATLAS Flight Tracker"
git push origin main
```

1. **Connect to Vercel:**
   - Go to vercel.com (https://vercel.com)
   - Import your repository
   - Configure project

2. **Project Settings:**

```
{
  "name": "3iatlas-flight-tracker",
  "framework": "vite",
  "buildCommand": "npm run build",
  "outputDirectory": "dist",
  "installCommand": "npm install"
}
```

1. **Environment Variables:**

```
# Add in Vercel dashboard
NEXT_PUBLIC_API_URL=https://api.3iatlas.com
```

vercel.json Configuration

```
{
  "buildCommand": "cd frontend && npm run build",
  "outputDirectory": "frontend/dist",
  "framework": "vite",
  "routes": [
    {
      "src": "/data/(.*)",
      "headers": {
        "Cache-Control": "public, max-age=604800, immutable"
      }
    }
  ]
}
```

Netlify Deployment

Option 1: Drag and Drop

1. Build locally:

```
cd frontend
npm run build
```

1. Go to netlify.com (https://netlify.com)
2. Drag `frontend/dist` folder to Netlify

Option 2: GitHub Integration

1. **netlify.toml Configuration:**

```toml
[build]
  command = "cd frontend && npm run build"
  publish = "frontend/dist"

[build.environment]
  NODE_VERSION = "18"

[[redirects]]
  from = "/*"
  to = "/index.html"
  status = 200

[[headers]]
  for = "/data/*"
  [headers.values]
    Cache-Control = "public, max-age=604800, immutable"

[[headers]]
  for = "/assets/*"
  [headers.values]
    Cache-Control = "public, max-age=31536000, immutable"
```

1. **Connect Repository:**
   - Go to Netlify dashboard
   - New site from Git
   - Select repository
   - Deploy

Option 3: Netlify CLI

```bash
# Install CLI
npm install -g netlify-cli

# Login
netlify login

# Initialize
netlify init

# Deploy
netlify deploy --prod
```

AWS S3 + CloudFront

Step 1: Create S3 Bucket

```bash
# Create bucket
aws s3 mb s3://3iatlas-flight-tracker

# Configure for static hosting
aws s3 website s3://3iatlas-flight-tracker \
  --index-document index.html \
  --error-document index.html
```

Step 2: Upload Built Files

```
cd frontend

# Build
npm run build

# Upload to S3
aws s3 sync dist/ s3://3iatlas-flight-tracker \
  --delete \
  --cache-control "public, max-age=31536000" \
  --exclude "*.html" \
  --exclude "data/*"

# Upload HTML with shorter cache
aws s3 sync dist/ s3://3iatlas-flight-tracker \
  --exclude "*" \
  --include "*.html" \
  --cache-control "public, max-age=3600"

# Upload data with medium cache
aws s3 sync dist/data/ s3://3iatlas-flight-tracker/data/ \
  --cache-control "public, max-age=604800"
```

Step 3: Create CloudFront Distribution

```
# Create distribution
aws cloudfront create-distribution \
  --origin-domain-name 3iatlas-flight-tracker.s3.amazonaws.com \
  --default-root-object index.html
```

CloudFront Distribution Config (JSON):

```
{
  "Comment": "3I/ATLAS Flight Tracker",
  "Origins": [{
    "DomainName": "3iatlas-flight-tracker.s3.amazonaws.com",
    "Id": "S3-3iatlas",
    "S3OriginConfig": {
      "OriginAccessIdentity": ""
    }
  }],
  "DefaultCacheBehavior": {
    "TargetOriginId": "S3-3iatlas",
    "ViewerProtocolPolicy": "redirect-to-https",
    "Compress": true,
    "MinTTL": 0,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000
  },
  "CacheBehaviors": [{
    "PathPattern": "/data/*",
    "TargetOriginId": "S3-3iatlas",
    "MinTTL": 604800,
    "DefaultTTL": 604800,
    "MaxTTL": 604800
  }],
  "Enabled": true
}
```

Step 4: Set Up Custom Domain

```
# Request SSL certificate
aws acm request-certificate \
  --domain-name tracker.3iatlas.com \
  --validation-method DNS

# Update CloudFront with custom domain
aws cloudfront update-distribution \
  --id E1234567890ABC \
  --aliases tracker.3iatlas.com
```

Next.js Deployment

If integrated into Next.js project:

Vercel (Recommended for Next.js)

```
# Automatic with git push
git push origin main

# Or manual
vercel --prod
```

Self-Hosted

```
# Build
npm run build

# Start production server
npm start

# Or with PM2
pm2 start npm --name "3iatlas-tracker" -- start
```

Docker (see Docker section)

Docker Deployment

Dockerfile

```
# frontend/Dockerfile
FROM node:18-alpine AS builder

WORKDIR /app
COPY package*.json ./
RUN npm ci
COPY . .
RUN npm run build

# Production stage
FROM nginx:alpine

COPY --from=builder /app/dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

nginx.conf

```
server {
    listen 80;
    server_name _;
    root /usr/share/nginx/html;
    index index.html;

    # Enable gzip
    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml ap
plication/xml;

    # Cache static assets
    location /assets/ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # Cache data files
    location /data/ {
        expires 7d;
        add_header Cache-Control "public";
    }

    # SPA fallback
    location / {
        try_files $uri $uri/ /index.html;
    }
}
```

Build and Run

```
# Build image
docker build -t 3iatlas-tracker:latest -f frontend/Dockerfile frontend/

# Run container
docker run -d -p 80:80 --name 3iatlas-tracker 3iatlas-tracker:latest

# Or with docker-compose
```

docker-compose.yml

```yaml
version: '3.8'

services:
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "80:80"
    restart: unless-stopped
    environment:
      - NODE_ENV=production

  # Optional: Add backend service for data updates
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    volumes:
      - ./frontend/public/data:/app/data
    environment:
      - PYTHONUNBUFFERED=1
    command: >
      sh -c "
        python generate_atlas_trajectory.py &&
        sleep 43200 &&
        python generate_atlas_trajectory.py --poll
      "
```

Performance Optimization

1. Data File Optimization

```bash
# Minify JSON
npm install -g json-minify
json-minify frontend/public/data/trajectory_static.json > trajectory_static.min.json

# Enable compression
gzip -9 trajectory_static.json
```

2. Asset Optimization

```bash
# Optimize images
npm install -g imagemin-cli
imagemin docs/*.png --out-dir=frontend/public/images

# Generate WebP versions
for f in frontend/public/images/*.png; do
  cwebp -q 80 "$f" -o "${f%.png}.webp"
done
```

3. Code Splitting

Already configured in Vite, but ensure:

```
// Dynamic imports for heavy components
const HeavyComponent = lazy(() => import('./HeavyComponent'));
```

4. CDN Configuration

For data files, consider using a CDN:

```
// Update data URLs to CDN
const DATA_CDN = 'https://cdn.3iatlas.com/data';

fetch(`${DATA_CDN}/trajectory_static.json`)
  .then(res => res.json())
  .then(setData);
```

Monitoring

1. Error Tracking (Sentry)

```
npm install @sentry/react @sentry/tracing
```

```
// main.tsx
import * as Sentry from "@sentry/react";

Sentry.init({
  dsn: "YOUR_SENTRY_DSN",
  integrations: [new Sentry.BrowserTracing()],
  tracesSampleRate: 1.0,
});
```

2. Analytics (Google Analytics)

```
<!-- index.html -->
<script async src="https://www.googletagmanager.com/gtag/js?id=G-XXXXXXXXXX"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());
  gtag('config', 'G-XXXXXXXXXX');
</script>
```

3. Performance Monitoring

```
// Monitor FPS and performance
import { useFrame } from '@react-three/fiber';

function PerformanceMonitor() {
  useFrame((state) => {
    const fps = Math.round(1 / state.clock.getDelta());

    if (fps < 30) {
      console.warn('Low FPS detected:', fps);
      // Report to monitoring service
    }
  });

  return null;
}
```

4. Uptime Monitoring

Use services like:
- UptimeRobot
- Pingdom
- StatusCake

Configure alerts for:
- Site downtime
- Slow response times
- SSL certificate expiration

Continuous Deployment

GitHub Actions Workflow

```yaml
# .github/workflows/deploy.yml
name: Deploy to Production

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Generate Trajectory Data
        run: |
          cd backend
          python3 generate_atlas_trajectory.py

      - name: Install Dependencies
        run: |
          cd frontend
          npm ci

      - name: Build
        run: |
          cd frontend
          npm run build

      - name: Deploy to Vercel
        uses: amondnet/vercel-action@v20
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-org-id: ${{ secrets.VERCEL_ORG_ID }}
          vercel-project-id: ${{ secrets.VERCEL_PROJECT_ID }}
          working-directory: frontend
```

Post-Deployment Checklist

- [ ] Verify site is accessible
- [ ] Test all interactive features
- [ ] Check mobile responsiveness
- [ ] Verify data loading
- [ ] Test playback controls
- [ ] Verify timeline events
- [ ] Check educational content display

- [ ] Test camera modes
- [ ] Verify performance (FPS)
- [ ] Check console for errors
- [ ] Test on multiple browsers
- [ ] Verify SSL certificate
- [ ] Set up monitoring alerts
- [ ] Update documentation

Rollback Plan

If deployment fails:

Vercel

```
vercel rollback <deployment-url>
```

Netlify

```
netlify rollback
```

AWS S3

```
# Restore from backup
aws s3 sync s3://3iatlas-backup/ s3://3iatlas-flight-tracker/
```

Docker

```
# Rollback to previous image
docker stop 3iatlas-tracker
docker run -d -p 80:80 --name 3iatlas-tracker 3iatlas-tracker:v1.0.0
```

Support

For deployment issues:
1. Check deployment logs
2. Review this guide
3. Open GitHub issue
4. Contact DevOps team

Happy deploying! 🚀