High Availability & Deployment

Overview

This document describes the deployment architecture and high availability measures for Spotify Genre Sorter.

Deployment Architecture

Blue-Green Deployment

The deployment pipeline uses a blue-green strategy with health gates:

```
┌─────────────┐    ┌─────────────┐    ┌──────────────┐    ┌─────────────┐
│  Quality    │───▶│  Staging    │───▶│ Production   │───▶│ DNS Cutover │
│  Checks     │    │  Deploy     │    │   Deploy     │    │             │
└─────────────┘    └─────────────┘    └──────────────┘    └─────────────┘
       │                  │                  │                   │
       ▼                  ▼                  ▼                   ▼
  ESLint, TS,       Health Check       Health Check        Update DNS
  Build, Audit      (3 retries)        (5 retries)         if healthy
```

Pipeline Stages

Stage	Duration	Failure Action
Quality Checks	~2 min	Block deployment
Security Scan	~1 min	Continue (warning)
Staging Deploy	~1 min	Block production
Staging Health	~15 sec	Block production
Production Deploy	~1 min	Abort
Production Health	~20 sec	Abort
DNS Cutover	~30 sec	Skip (warning)
Cleanup	~30 sec	Skip (warning)

Health Check Implementation

Staging Health Check:

```
for i in 1 2 3; do
  curl -s https://staging.workers.dev/health
  # Retry 3 times with 5 second delay
done
```

Production Health Check:

```
for i in 1 2 3 4 5; do
  curl -s https://production.workers.dev/health
  # Retry 5 times with 5 second delay
done
```

Zero-Downtime Guarantees

Why Cloudflare Workers Provide Zero-Downtime

1. **Atomic Deployments**: New code is uploaded alongside existing code
2. **Global Edge**: Changes propagate to all edge locations
3. **No Container Restarts**: V8 isolates start instantly
4. **Shared KV**: Both versions share the same KV namespace

Deployment Overlap Window

```
Old Version ────────────┐
                        │ (Requests routed to either)
New Version ────────────┴────────────────────────────▶
            ^                        ^
         Deploy Start           Propagation Complete
            (~1 min)                 (~30 sec)
```

During this window (~90 seconds):

- Some requests may hit old version
- Some requests may hit new version
- Both versions use same KV data

KV Schema Compatibility

Guidelines for schema changes:

1. **Additive changes only**: Add new fields, don't remove
2. **Default values**: New code handles missing fields
3. **Version fields**: Include schema version in cached data

Example backward-compatible code:

```
// Handle both old and new cache format
const cache = await kv.get(key);
if (cache) {
  const data = JSON.parse(cache);
  // Default for new field if missing
  const truncated = data.truncated ?? false;
}
```

Graceful Degradation

Spotify API Failures

Scenario	Behaviour
429 Rate Limited	Retry with Retry-After header
500/502/503	Retry with exponential backoff (3 attempts)
Network error	Retry with backoff
Complete outage	Return error with step info

Error Response Example:

```json
{
  "error": "Failed to fetch artist data from Spotify",
  "details": "429: Too Many Requests",
  "step": "fetching_artists",
  "tracksFound": 500,
  "artistsToFetch": 200
}
```

KV Failures

Scenario	Behaviour
Read fails	Continue without cache
Write fails	Continue, log error
Complete outage	Sessions fail (401)

Session Degradation

If session cannot be read:

1. User gets 401 Unauthorized
2. Frontend redirects to login
3. New session created on login

Rollback Procedures

Automatic Rollback

No automatic rollback implemented. Cloudflare Workers deployments are reliable enough that manual intervention is rare.

Manual Rollback

Option 1: Redeploy Previous Commit

```
git checkout HEAD~1
git push -f origin main
# Wait for deployment
```

Option 2: Cloudflare Dashboard

1. Go to Workers & Pages
2. Select worker
3. Go to Deployments
4. Click "Rollback" on previous deployment

Option 3: Wrangler CLI

```
# List deployments
npx wrangler deployments list

# Rollback to specific deployment
npx wrangler rollback <deployment-id>
```

Rollback Time Estimates

Method	Time to Complete
Cloudflare Dashboard	~30 seconds
Wrangler CLI	~30 seconds
Git push redeploy	~5 minutes

DNS Configuration

TTL Settings

Record Type	TTL	Purpose
CNAME	1 (auto)	Fast propagation
Workers route	Instant	Edge routing

DNS Propagation

Cloudflare proxied records propagate in <5 minutes globally. Custom domains use Workers Custom Domains API for instant edge routing.

Monitoring During Deployment

GitHub Actions View

The deployment creates a summary report:

```
# Deployment Report

| Stage | Status |
|-------|--------|
| Code Quality | success |
| Security | success |
| Staging | success |
| Production | success |
| DNS Cutover | success |
```

Health Endpoint

Check production health:

```
# Basic check
curl https://production.workers.dev/health
# Returns: {"status":"ok","version":"1.2.1"}

# Detailed check
curl https://production.workers.dev/health?detailed=true
# Returns: {"status":"ok","version":"1.2.1","components":{...}}
```

Incident Response

Runbook: Service Down

1. **Verify the issue**

   ```
   curl -v https://swedify.houstons.tech/health
   ```

2. **Check Cloudflare Status**
   - https://www.cloudflarestatus.com/
3. **Check GitHub Actions**
   - Recent deployments that may have broken things
4. **Rollback if needed**
   - Use Cloudflare dashboard for fastest rollback
5. **Check logs**

   ```
   npx wrangler tail spotify-genre-sorter
   ```

Runbook: High Error Rate

1. **Check Spotify API Status**
   - https://developer.spotify.com/status
2. **Check rate limits**
   - Look for 429 responses in logs
3. **Check KV availability**
   - Try /health?detailed=true endpoint

Runbook: DNS Issues

1. **Verify DNS resolution**

   ```
   nslookup swedify.houstons.tech 8.8.8.8
   ```

2. **Check Cloudflare DNS settings**

   - Verify CNAME record exists
   - Verify proxy is enabled (orange cloud)

3. **Force DNS update**

   - Re-run workflow with "Force DNS update" checked

SLA Targets

Metric	Target
Uptime	99.9% (Cloudflare SLA)
Deploy time	<5 minutes
Rollback time	<1 minute
Health check	<1 second
Time to first byte	<100ms

Related Documentation

- Security Architecture
- Operational Efficiency
- API Documentation

Last updated: December 2025